Introduction aux Base de Données et ACCESS

PREMISE

Ce manuel (ou syllabus) ne substitue pas la documentation officielle d'ACCESS™, il est simplement une trace pour apprendre à utiliser ce produite. Pour suivre efficacement ce cours élèves et utilisateurs devraient déjà avoir une connaissance de base de l'ordinateur et d'OFFICE™.

Entre différentes versions d'ACCESS il y a des petites différences sur la nomenclature, les noms de certains attributs, voir dans l'aide le nom anglais.

CONVENCTIONS

La police Verdana sera utilisé pour indiquer les affichage du menu, ex: Tableau, Propriétés du Tableau.

La syntaxe des commands, les instructions en langage de programmation et les exemples sont avec police `COURIER NEW`. Dans la syntaxe les parties optionnelles sont renfermées entre `[parenthèses carrées]` et les mots variables sont en *`italique`*.

1. BASE DE DONNEES

1.1 HISTOIRE

Le traitement des informations s'est évolué dans les temps, suivant le développement de la technologie, qu'a mis à disposition capacités de mémorisation et puissance de calcul toujours croissantes. La table suivante est une synthèses de cette évolution.

AN	ORGANIZATION	FUNCTIONNALITE'	PRODUITS [1]	SUPPORTS PHISIQUES
1950	**Fichiers**	Access sequentielle		Cartes perforées, rubans magnétiques
1960	**Fichiers avec Indexes**	Indexe	ISAM, VSAM	Disques
1970	**Banque de données (Base de données)**	Indexe, Description des données, langage de programmation	DBASE, CLIPPER, FOXPRO	Disques
1980	**Banque de données**	Archive avec données, Indexes, description des données, langage de programmation	ACCESS	Disques
1990	**Serveur pour Banque de données (*DBMS: Base de données Management System*)**	Archive avec données, Indexes, description des données, langage de programmation, exécution des logiciels par le Base de données	ORACLE, DB2, SQL SERVER, INGRES, ...	Données distribués sur disques en LAN et WAN
future	**SO + DBMS**			

Figure 1

En même temps aux développements de la technologies, se sont posé les bases théorique du traitement des informations, qui on porté au model relationnel des données.

Les données, dans leur forme élémentaire, sont une couple nom attribut[2] par exemple:

ville, "BAMAKO",
bitmap, "*le condor du logo*"

ou même nom attributs :

Montaigne, "MONT BLANC",Hauteur,4808.[3]

En général les données sont groupé pour former des unités logiques, par exemple l'ensemble des données anagraphiques, des attributs d'une image, ou les caractéristiques d'une montaigne. Plusieurs informations élémentaires agrégées sont dites génériquement *entité*, parfois record, fiche ou en terminologie relationnel *lignes* (*Row*). Les entité à sont tour sont agrégées en ensembles dits files ou fichier ou archive ou *Tables*. Les Base de Données, en anglais Base de données (DB), enfin, sont l'ensemble de plusieurs tables et d'autres informations corrélés. Une donnée générique de la ligne est dite aussi *champ* (*field*).

L'ordinateur permet de traiter les informations de façon beaucoup plus flexible et rapide de l'élaboration manuel; l'ensemble des tables, des informations corrélés et d'outils informatiques de gestion comme la gestion automatique des indexes, la sûreté, la programmabilité, etc. il est dit DBMS (*Data Base Management System*).

Les agrégés d'informations peuvent avoir aussi structures complexes, par exemple la fiche d'un employé peut contenir des sous fiches relative aux augmentations qu'il a eu; si on essaye de faire un schéma de cette structures on obtienne des graphes à arbre:

[1] Les produits cités sont seulement aucunes, parmi les plus significatifs, qui ont marqué l'histoire du traitement des données.
[2] Aussi une couple substantif -adjectif ou substantif -nom propre.
[3] Dans la mémoire de l'ordinateur ou sur le support de mémorisation existe seulement l'attribut; le nom correspond à l'adresse ou à la position du champ dans la structure que le contienne.

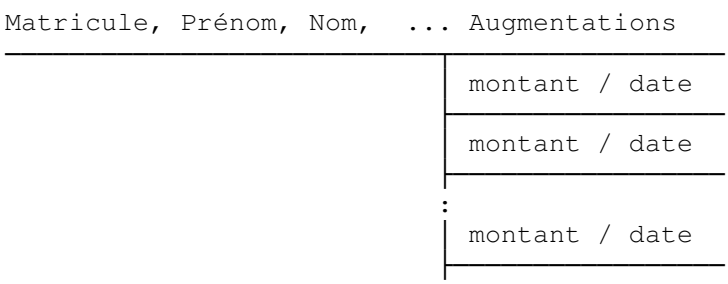

Figure 2 ***Un ensemble hiérarchique***

Les Base de données qui acceptent structures complexes sont dits hiérarchiques ou réticulaires. Puisque l'usage des DB hiérarchiques est complexe, ils ont été remplacés par les Base de Données Relationnels qui contiennent une ou plusieurs tables dans les quelles les colonnes sont les champs, et le lignes sont les fiches. La simplicité de la structure relationnelle, d'autre partie, veut une certaine redondance représentée par un champ en commune parmi certaines tables. La représentation relationnelle des données de l'exemple de la Figure **2** origines les tables anagraphique et la table augmentation; la colonne commune est la matricule.

Prénom	Nom	Matricule	.....
ROSSI	JEANNE	805567	...
BIANCHI	ETIENNE	231784	...
VERDI	PAUL	901526	...

Figure3 ***Table Anagraphique***

Matricule	Montant	Date	.....
805567	30	31/08/1998	...
231784	20	30/04/1996	...
901526	40	30/04/1999	...
231784	30	30/04/2000	...

Figure 4 ***Table Augmentations***

La simplicité conceptuelle des DB Relationnels est évidente dans les opérations que on peut faire sur eux à l'aide d'un langage dit SQL (*Structured Query Lanquage*).

Avant de créer un logiciel ou une application[4], on doit faire une activité beaucoup importante, c'est à dire penser à ce que on veut obtenir et à les informations dont on à besoin. Cette activité est dite *analyse*; en particulier il faut déterminer les données qui sont nécessaire, le type d'eux (alphanumériques, nombres, date, etc.), leurs dimension, les valeurs acceptables (*domaine*) et les relations entre eux.

Nom de l'attribut	Type	Dimension	Domaine
NOM	Caractères	30	
DATE DE NAISSANCE	Date		
LIEU DE NAISSANCE	Caractères	25	
SALAIRE	Nombre		> 20000 et < 100000
...			
SEXE	Caractères	1	M ou F

Figure 5

[4] Une application est un ensemble de logiciels pour la gestion d'activités complexes: par exemple la gestion du personnel, la facturation, le contrôle d'un magasin, etc..

1.2 LES TABLES RELATIONNELLES

1.2.1 DEFINITIONS

Le modèle des base de données relationnels a été introduite par ET. F. Codd, en IBM[5], qu'il a formulé le modèle mathématique et il a spécifié un langage d'interrogation. A' partir de son travail il est né le DB2, la première base de données relationnelle commerciale. Ici suivent quelques définitions.

- Les *entités* sont déterminées par un ensemble finit de couples `<attribut, valeur>` dit aussi *tuple*, mais il est normalement employé le mot *ligne* (*row*).
- L'ensemble des attributs est dit *schéma*.
- L'ensemble des possibles valeurs des attributs est dit *domaine*
- *Relation* est l'ensemble des tuple qu'ils ont le même schème, normalement on utilise le synonyme *table*.

En se rapportant à a Figure 6, dans la quelle la colonne *nom* est le *schéma* de la Relation Employés, une possible entité pourrais être:

```
NOM                     JEAN VALJEAN
DATE DE NAISSANCE       31/10/1960
LIEU DE NAISSANCE       JEUVILLE
SALAIRE                 5000
...                     ...
SEXE                    M
```

Figure 6

Le schéma est: {`NOM, DATE DE NAISSANCE, LIEU DE NAISSANCE, SALAIRE,..., SEXE`}, le domaine de l'attribut `SEXE` est {`M, F`}.

On dit *clé primaire* (*primary key*) un ensemble minimum d'attributs, s'il existe, qui ont valeurs univoques, dans l'exemple la clé primaire pourrais être {`NOM, DATE DE NAISSANCE`}, parce que dans la Relation on peut avoir deux "`ROUGE PIERRE`", avec `DATE DE NAISSANCE` différentes.
Autres éventuelles clés sont dites *clés secondaires*.
La diction *relationnel* dérive de la possibilité de mettre en relation entre eux des tables différentes, en fonction d'attributs communs, dans la table dessous l'attribut `CodeClient` met en relation les tables Clients et Ventes, l'attribut `CodeMatériel` lie les tables Ventes et Matériels.

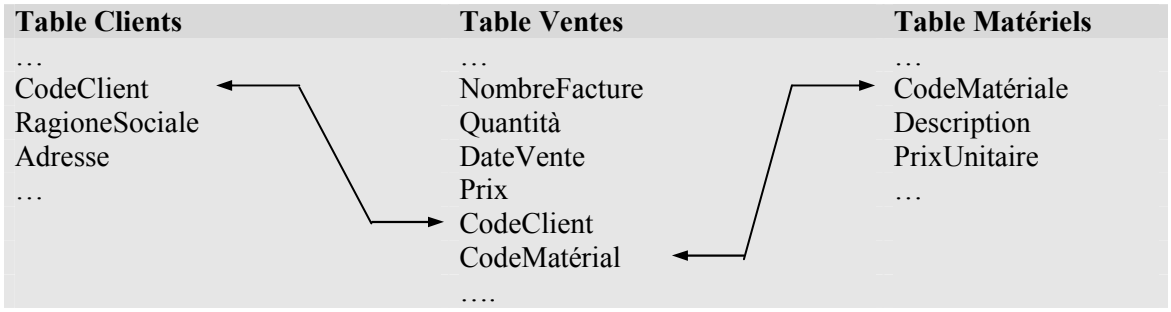

Figure 7

1.2.2 OPERATIONS

L'opération de *Sélection* permet de "extraire" des entités de une Relation par un prédicat sur les attributs de la relation, par exemple la *Sélection* "`SALAIRE > 5000`" sur la Relation Employés, produise une nouvelle relation qui contienne seulement entités dans les quelles l'attribut `SALAIRE` vaut plus de `5000`.

Entre Relations ayant le même schéma sont possibles les opérations sur ensembles *Union*, *Intersection*, et *Différence*.

Avec *Projection* d'une relation *R* on entende la relation *R'* obtenue de un sous ensemble *S* des attributs de *R*.

[5] ET. F. Codd "A Relational Model of Data for large Shared Data Banks," by Dr. ET. F. Codd, dans "the Association of Computer Machinery (ACM) journal, Communications of the ACM" June 1970.

Le *Produit* de deux relations R_1 et R_2 (il est dit aussi *Produit Cartésien*) est une relation dont le schéma est l'union des schémas de R_1 et R_2 et qui contienne l'union de chaque tuple de R_1 avec chaque tuple de R_2.

Une *Sélection* sur le *Produit* de deux relations R_1 et R_2, obtenue par un prédicat de égalité parmi les attributs A_1 de R_1 et A_2 de R_2, (attributs ayants le même domaine), est dite *equi-conjonction* ou *equi-join*, et le résultat est une relation contenant les tuple de R_1 unies avec les tuple de R_2 pour le quelles vaut $A_1 = A_2$.

1.3 LES DBMS

Avec le mot DBMS on indique un logiciel qui gère des Base de Données en modalité Client Server c'est-à-dire exécutant des requêtes de données ou de modification de la Base de Données, provenant de plusieurs Clients en même temps.

1.3.1 STRUCTURE PHYSIQUE

La structure physique d'une DBMS varie selon le spécifique Base de Données : dans les paragraphes qui suivent il y a quelques exemples.

1.3.1.1 ACCESS

ACCESS techniquement n'est pas un vrai DBMS, même s'il a des aspects d'un DBMS.
ACCESS est en général une Base de Données contenue dans un seul fichier, avec l'application pour la gérer, c'est à dire le fichier peut contenir tables, formulaires, logiciels etc..

1.3.1.2 MySQL

MySQL est un DBMS gratuite, moyennement puissante et beaucoup utilisé pour des applications Internet.

Le Base de Données MyISAM correspondent à un répertoire du disque, chaque table est composée par un fichier de données, un fichier de description de la table et éventuel fichier avec les indexes. Les fichiers ont des noms formé par le nom de la table et les extension sont : `.frm` contient le format de la table, `.MYD` est le fichier de données, `.MYI` est le fichier des index.

1.3.1.3 Oracle

ORACLE est un des plus puissante DBMS, sa structure est basé sur :

- un ou plusieurs fichiers de données *(Data Files)*: ils contiennent les données et tous les informations relatives à leur élaboration,
- un ou plusieurs fichiers de log *(Redo Logfiles)*: ils contiennent l'histoire des modifications faites au Base de Données ; ils sont nécessaires pour éventuels reconstructions de la Base de Données dans le cas de fautes ou perte des données,
- un ou plusieurs fichiers de contrôle (*Control Files*): ils contiennent des informations générales sur la Base de Données : nom, place des archives, etc..

1.3.2 STRUCTURE LOGIQUE

Un DBMS on peut avoir un ou plusieurs Base de données (dans la terminologie Oracle *TableSpace*), et dans eux des objets, *Tables*, mais aussi *Requêtes*, *Indexes*, *Logiciels*, *Conteurs*, …
Le *requêtes* ou *Views* sont tables "virtuelles", dans le sens que dans le DBMS est mémorisé comme obtenir les données d'une ou plusieurs tables ou vues mêmes. Comme les tables, les vues peuvent être interrogées et les données modifiées et ou effacées.
On utilise les vues pour:

- Cacher la complexité de trouver des données qui se trouvent en tables différentes.
- Présenter les informations dans une différente perspective égard à celle de la table réel.
- Ajouter un ultérieur niveau de sûreté, en limitant l'accès à prédéterminées ensembles de lignes et de colonnes d'une ou de plusieurs tables.

Les *Indexes* sont la composante de Banques de Données qui permettent l'accès rapide à des informations.
Les Logiciels sont la composante "personnalisé" du DBMS, ils sont fonctions, procédure ou élaborations que le DBMS effectue au vérifier d'évents particuliers (*trigger*).
Il est à remarquer que pratiquement toutes les informations qui servent au DBMS pour sa gestion sont maintenues dans des tables d'une Base de données de système.

1.3.3 LE MOTEUR DU DBMS

Les DBMS ne sont pas simplement des fichiers d'informations, mais ils sont des gérants d'informations: dans un ambiante multi usagers ils permettent l'accès en contemporanéité aux données en assurant un haute niveau de prestation, l'intégrité des données et des solutions efficaces pour faire front aux fautes du à dégât physiques. Les DBMS exécutent aussi des taches prédéterminées et automatiques, comme le journal des activités, la gestion des indexes et des espaces, l'exécution d'instructions liées à certaines évents (*trigger*, fermeture du travail…), etc..

1.4 LE "LANGAGE" SQL

SQL est devenu le standard pour accéder aux données contenues dans les Base de données relationnels. Techniquement SQL il n'est pas un langage de programmation complet, ils manquent en effet des instructions pour conditionner l'exécution. Il est surtout un outil pour interagir avec un DB relationnel à niveau logique, avec une haute abstraction de l'implémentation des données. Entre les possibilités de SQL on a:

- création, modification et effacement d'objets,
- interrogation de données, le résultat est une relation,
- insertion, modification et effacement de lignes (*tuple*).

Ici il y a seulement un aperçue de SQL selon la syntaxe accepté par ACCESS.

1.4.1 CREATION D'UNE TABLE

Pendant la création d'une table, outre à déterminer les champs et leur format, il est possible indiquer si celui ci est une clé et de quel type.

ACCESS (et les DBMS) acceptent beaucoup type de donnés, ici une petite liste des plus communes :

Type	Description
`INTEGER, LONG`	Pour mémoriser des nombre entier
`MONEY`	Pour mémoriser des nombre avec quatre décimaux
`AUTOINCREMENT`	Nombre entier qui se auto incrément pour chaque nouveau ligne
`VARCHAR,TEXT`	Pour les champs de texte jusqu à 255 caractères, ex. `TEXT(12)`
`DATE`	Pour mémoriser des dates et heures
`MEMO`	Pour mémoriser texte de longueur supérieure à 255

`Ex. :`

```
CREATE TABLE anag
      (Nom       VARCHAR(30) PRIMARY KEY,
      Adresse    VARCHAR(30),
      CAP        VARCHAR(5),
      Quantité   DOUBLE,
      Ville      VARCHAR(25) NOT NULL,
      Province   VARCHAR(2))
```

`PRIMARY KEY` engendre un index univoque et pas nulle sur la colonne *nom*. L'indication **`NOT NULL`** de la colonne *province* indique que la colonne doit exister.

Pour modifier une table le command est `ALTER TABLE` *`nomdetable ...`*:

Exemples :

- Ajouter un champ : `ALTER TABLE clients` **`ADD`** `COLUMN Telephone TEXT(20)`
- Modifier un champ : `ALTER TABLE clients` **`MODIFY`** `COLUMN Nom TEXT(30) NOT NULL`
- Effacer un champ : `ALTER TABLE clients` **`DROP`** `COLUMN Province`

L'instructions pour éliminer une table est : `DROP TABLE` *`nomdetable`*

1.4.2 LES INTERROGATIONS

La figure suivante montre la structure du commando `SELECT` de SQL qui opère sur une ou plusieurs relations:

SELECT	Schema	proiection
FROM	Relation(s)	Introduise la(les) relation(s)
WHERE	Condition(s)	Sélection ou join
GROUP BY	Attribut(s)	Groupement d'attributs
ORDER BY	Attribut(s)	

Figura 8

Seulement les clause `SELECT` et `FROM` sono nécessaires.

Le command SELECT est suivi des données que on veut voir ; celui-ci peut être :

- `*` tous les champs,
- une liste de champs et/ou de fonctions sur eux, séparées par `,` (virgule).

Le commande visualise les données voulues avec un entête formé par le nom du champ, toutefois est possible indiquer un autre nom, surtout pour les champs calculées, avec la syntaxe suivante :

```
SELECT ... nom AS entête, ...
```

Tous les champs	`*`
Quelques champs	`Nom, Adresse, Ville, ...`
Valeurs calculés	`..., Montant * 1,20, ...`
Champs avec entête de l'usager	`..., Montant * 1.20 AS [MONTANT + IMPOT], ...`

`Exemples :`

- `SELECT * FROM anag`
- `SELECT nom, adresse, Montant * 1.20 AS [Montant lourd], Préfixe + '/' + téléphone AS télé FROM anag`

1.4.2.1 La clause FROM

`FROM` indique la table ou les tables dont on veut les données. La table peut être aussi engendré par un command `SELECT`. La syntaxe est :

`SELECT ... FROM` *relation*$_1$`[AS` *alias*$_1$`],`*relation*$_2$`[AS` *alias*$_2$`],...`

Où *relation*$_n$ est le nom d'une table ou une `SELECT`.

Quand il y a plusieurs tables on peut indiquer champs de tous les tables. Dans la liste des champs de SELECT les ambiguïtés (nom de champs égales dans les différentes tables), sont évité par la syntaxe :

`SELECT ...,`*nomdetable.nomdechamp*`, ... FROM ...,`*nomdetable*`,...`

Où *nomdetable* est le nom de la table ou le nom assigné par la clause `AS` *alias*$_n$.

Exemples

- `SELECT nom, anag.province, chieflieu FROM anag, tab_pro`
- ```
  SELECT F.nom,Quantitè FROM farm AS F,
            (SELECT nom, sum(qty) AS Quantitè FROM Med GROUP By Nom) AS M
     WHERE F.nom = M.nom
  ```

1.4.2.2 La clause WHERE

Avec la clause `WHERE` on choisit les lignes de la table que on veut voir, la clause est suivie par une expression qui est évalué pour toutes les lignes de la table, si la relation est vraie la ligne sera utilisé.

- ```
  SELECT nom, anag.province, chieflieu FROM anag, tab_pro
  WHERE anag.province = Tab_pro.province
  ```

Dans expression les constants caractères sont entre apostrophes.

Les opérateurs de confrontation sont `=`, `>`, `<`, `>=`, `<=`, `LIKE`, `IN`. Les opérateurs logique `AND`, `OR` et `NOT` sont utilisable dans la condition :

- `condition`$_1$ `AND condition`$_2$ sera vraie si tous les deux conditions sont vrais
- `condition`$_1$ `OR condition`$_2$ sera vraie si au moins une condition est vraie
- `NOT condition` sera vraie si la condition est fausse

LIKE est utilisé pour des confrontations avec le contenu partielle des champs de texte, à l'aide de caractères de masque * et ? (*wild card*)[6] :

- * quelconques ensembles de caractères
- ? un caractère : ex. ?oss

ex. SELECT * FROM ANAG WHERE Nom LIKE '?oss*' trouve Rossati, Rossi, rossini, Fossati, etc.

Examples:

```
SELECT * FROM ANAG WHERE Province IN ('AL','NO')
SELECT * FROM ANAG WHERE Province NOT IN ('AL','NO')
SELECT * FROM ANAG WHERE Quantité > 4 AND Quantité  < 11
```

1.4.2.3 La clause GROUP BY

La clause GROUP BY ... sert pour obtenir des données synthétisées. Dans le groupement peuvent seulement paraître les champs sur les quels se font les groupements et/ou des fonctions d'agrégation : AVG, COUNT, MAX, MIN, et SUM.

- SELECT provincia, Count(Provincia) AS [N.],SUM(quantita) AS [TOTAL PROVINCE] FROM anag **GROUP BY** provincia ORDER BY Provincia;
- SELECT bureau, MIN(salaire), MAX(salaire) FROM emp GROUP BY bureau

```
PR         N. TOTAL PROVINCE
-- --------- --------------
CN          2            2.5
PD          1            2.7
RM          1            2.2
TO          1            1.5
VR          2            1.9
```

La liste aura implicitement l'ordre des champs de GROUP BY, a moins de la présence de la clause ORDER BY.

Il est possible choisir des données de la table obtenue, par la clause HAVING, qui est analogue à la clause WHERE.

Cet exemple au dessous montre comme ajouter un total à l'aide du command SQL UNION (UNION veut ensemble de données avec la même structure) :

```
SELECT Ville, SUM(Quantité) AS [Total Ville], COUNT(*) AS NOMBRE FROM ANAG
GROUP BY Ville
UNION
SELECT  'TOTAL',SUM(Quantité), COUNT(*) FROM ANAG
```

1.4.2.4 La clause ORDER BY

Pour obtenir les données ordonnées, même avec plusieurs champ ou expression, séparé par virgule. Le mot DESC après le nom indique un ordre descendant :

```
SELECT * FROM MED order by nom,qty DESC
```

nom	qty
Carbacolo Alfa	17
Carbaica	20
Carbaica	15
Debridat	70
Debridat	30

1.4.3 MODIFICATION DES DONNEES

1.4.3.1 Insertion de nouveaux données dans une table

```
1) INSERT INTO anag (nom, adresse, cap, province, ville)
       VALUES ('ROSSATI Giovanni', 'Oriani 3', '10149', 'TO','Torino');
2) INSERT INTO anag (nom, ville, province, cap)
       SELECT nom, ville, province, cap FROM anag2
       WHERE province = 'TO';
```

L'exemple 2) "copie" le colonne *nom*, *ville*, *province* de la table *anag2* appartenantes au lignes qui satisfint à : WHERE province = 'TO' et les insère dans la table *anag*.

1.4.3.2 Variation des données d'une table

```
1) UPDATE anag SET quantite = 1.5 WHERE nom like '?oss*'
2) UPDATE anag SET adresse = 'Castelrotto 66', CAP = '23023'
```

[6] Les *wild card* d'ACCESS ne sont pas standard, dans MySQL et Oracle sont respectivement % et _.

```
WHERE nom = 'Franchi Elmo';
```

1.4.3.3 Effacement des données d'une table

```
1) DELETE FROM anag WHERE nom = 'Fossati';
2) DELETE FROM anag;
```

L'exemple 2) efface toutes les données contenues dans la table *anag*. La table existe encore.

2. ACCESS PRATIQUE

Access est un logiciel pour mémoriser et élaborer informations structurées. Access aide l'utilisateur dans toutes les phases que regardent le traitement de données: du projet des tables, à la réalisation de formulaires pour la gestion des données, jusqu'à la préparation des états. Le langage Basic permet d'ajouter des fonctionnalités à l'application.
Beaucoup fonctions sont exécuté avec l'aide de « Assistants », et un aide en ligne est disponible dans tous contexte par la touche F1.

Les principaux objets d'ACCESS sont:

- Tables: sont la place où les information sont contenues; dans la définition des tables on détermine les informations contenues, avec ses caractéristiques, si il y a des contrôles sur les valeurs, l'éventuel valeur de défaut.
- Requêtes ou *query*: il est un façon de voir ensemble les données d'une ou plusieurs tables, par exemple la facture avec les données de la table des ventes, des clients et du magasin. Les données des requêtes peuvent aussi être le résultat d'opérations sur les données d'une table, par exemple le prix escompté.
- Formulaires: elles sont des interfaces graphiques pour visualiser les données des tables ou des requêtes et pour les gérer, c'est à dire pour modifier, effacer et insérer des nouveaux données.
- Etats ou *report* : elles sont la description pour obtenir l'états des données d'une table ou d'une requêtes, avec éventuels triage et totalisations.
- Modules: ils permettent d'ajouter des fonctions particulières aux fonctions natives d'Access. Les modules sont écrits avec le langage de programmation VBA (*Visual Basic for Applications*), qu'il est aussi utilisé pour gérer les évents des formulaires, et des états.

Au démarre de Access, il propose:

- Nouvelle base de donnée Accces,
- Assistant, pages ... pour choisir entre diverses projet-example
- Ouvrir un fichier existant.

Ils existent en ACCESS deux modalité de travail : *exécution*, qui permet de gérer des données, et *projet* qui permet de gérer les composants de la Base des données, c'est-à-dire tables, requêtes (*query*), formulaires pour l'introduction des données, brouillon pour les états et programmation. Dans ce chapitre on examinera la modalité *projet* d'ACCESS.

La création d'une application est géré par des formulaires et pour la plus part des composant de ACCESS est disponible une construction conduite (Assistant).

Il y a une fenêtre pour gérer les objet d'ACCESS, choisissant un objet il est possible choisir entre la création d'un nouveau ou la modification.

2.1 TABLES

Il est possible créer un nouveau table, soit en indiquant les champs dans la fonction Visualisation structure, soit en copiant, et éventuellement en modifiant, une structure de table préexistant.

2.1.1 TYPE DES CHAMPS

En Visualisation structure il faut indiquer obligatoirement les noms des champs et leur type. Cette dernière indication est choisie par un menu à descente qu'il est accessible en cliquant sur le champ Type de donnée. Ici en bas une aperçue des type de champs.

type	utilisé pour	caractéristiques
texte	adresses, noms, codes, même si numériques, ...	de 0 aux 255 caractères
numéro	montants, quantité, pourcentages, …	entiers ou avec des chiffres décimaux
date/heure	dates et heures	
compteur	matricules, codes clients, matériels, factures.	le champ est géré automatiquement et il sert à numéroter les fiches
mémo	textes, documents, images.	
oui/non	oui/non, vrai/faux, absent/présent.	

Table 2-1 Type des champs

Au-delà de nom et type, il est possible d'indiquer autres attributs du champ, ceux-ci servent à faciliter l'introduction et le contrôle des données. Ces attributs dépendent du type de donnée.

Quand un des attributs est sélectionné, il peut apparaître à droite un bouton de liste pour permettre un choix entre différentes alternatives ou un bouton avec `. . .` qui fait apparaître une fenêtre de dialogue.

attribut	utilisé pour
Dimension	Indiquer la dimension maxime d'un champ texte ou numérique
Forme	presque pour toutes les types, utile pour dates et les montants
Masque d'input	Pour guider l'introduction des données
Étiquette	Aide pour l'input des données
Valeur prédéfinie	Pour introduire une valeur prédéfinie
Valide si	indique un contrôle formel
Message pour faute	Utilisé pour signaler l'introduction de données erronées
Obligatoire	Impose l'introduction du champ
Permets longueur zéro	
Indexé	Cree un indexe pour rendre vites les recherches et les imprimeries

Table 2-2 Attributs des champs

Remarque Le champ `Etiquette` est utilisé pour donner l'étiquette aux champs des formulaires et des états, ceci permet de développer vitement et utiliser des dictions homogènes.

2.1.2 CONTROLES SUR LES CHAMPS

Dans les champs Valide si il est possible d'insérer une expression de contrôle direct (voir Table 2-3 **Exemple de contrôles**) ou une expression avec des fonctions de *VBA* ou des fonctions de contrôle écrites avec *VBA* par le programmateur L'expression de contrôle peut être composée avec l'aide du "Générateur d'expressions", ceci on obtient avec click sur le bouton qui apparaît à droite du champ "Valide si", quand cela est sélectionné.

Un éventuel contrôle d'unicité on l'obtient en indiquant le champ comme indexé et en choisissant l'option Oui duplicatures pas admis.

`Date()+10`	Date du jour plus 10
`>= 65`	Valeur majeur ou égale de 65
`LIKE 'ROSS*'`	Accepte noms qui commencent par ROSS
`'M' Or 'F'`	Accepte seulement M ou F

Table 2-3 Exemple de contrôles

2.1.3 LES INDEXES

Les index sont nécessaires pour grandes tables, il est possible d'introduire index formé par plusieurs champs et même sur des expressions, par la fonction de menu Affichage et puis Index. La balise avec la clé  indique Primary key c'est à dire un index qu'il doit être présent et univoque**.**

2.2 FORMULAIRES

Les Formulaires sont les objets qui permettent de gérer les données par introduction guidée. ACCESS permet de créer des formulaires personnalisés ou avec une Création guidée. Les formulaires sont associés, normalement, à un table ou à un requête. Les formulaires si obtenue sont douées par défaut de buttons pour créer, modifier, effacer et rechercher des donnés.

2.2.1 LES OBJETS DES FORMULAIRES

Les *formulaires* sont des conteneurs d'objet pour insérer de données (zone de texte, zone de liste, zone de liste modifiable ou *combo box*), ou pour les gérer (buttons, radio buttons, check box, barre di déroulement, etc..) ou pour obtenir des effets esthétique (images, lignes, carres, etc.).
A chaque objet sont associé des propriétés qui sont accédé par :

- Double clique sur l'objet,
- Clique droit et choix Propriété.
- Balise des propriétés :  .

2.2.2 CREATION D'UN FORMULAIRE

On peut engendrer un formulaire a l'aide de Access (Créer un formulaire a l'aide l'Assistant) ou le bâtir soi même déplaçant les objet sur la fenêtre. Les objets sont choisis dans la Boite a outils (Menu Affichage, Boite a outils) ou la balise  .

- Texte est utilisées pour insérer du texte ou visualiser des informations, les textes peuvent être associés à des champs d'une table ou être engendré par une formule.
- Listes et combo pour choisir des données entre un ensemble fixe de valeurs, le combo peut aussi permettre l'introduction d'un texte pas présente dans la liste (propriété Limiter à list No). Si dans la boite des outils la balise  est choisie, une procédure guidée permettra d'indiquer si les données sont dans une Table ou sont obtenue par une Requête ou sont indiqué directement.
- Buttons pour exécuter des commands, si la balise  est active, on est conduite sur des choix entre plusieurs actions et il est automatiquement engendré le code VBA pour la gestion.
- Check Box et Option Box

L'Assistant engendre un formulaire ou le champs correspondent a zone de texte, sauf le données de type Oui/Non qui sont visualise par des check box.

2.2.3 PROPRIETE ET EVENEMENTS DES FORMULAIRES

On accède aux propriétés et événements des formulaires en sélectionnant le formulaire avec un click sur le bouton en haut à gauche au croisement des règles. Ici de suite quelques propriétés.

propriété	valeur	caractéristiques
Étiquette	texte	titre du formulaire
Style du borde (BorderStyle)	numéro	il détermine aspect et fonctionnalité du formulaire (dialogue) popup, redimensionnable,
Boutons de déplacement	si/no	pour se déplacer, en avant, en arrière, début, fin, nouveau record,
Bouton de fermeture	si/no	
Image	bitmap	fond du masque
Origine fiches	texte	le nom d'un table ou d'un requête+ ou une instruction SQL SELECT
Trie pour	texte	le nom d'un champ

Ici de suite quelques événements associé au formulaires :

événement	caractéristiques
Sur chargement (onLoad)	pour fonder des propriétés du formulaire avant qu'il soit visualisé
Sur ouverture (onOpen)	pour opérations précédentes la première visualisation de données
Sur activation (onCurrent)	avant de chaque visualisation des données
Sur fermeture (onClose)	pour les opérations à la fermeture du formulaire

2.2.4 PERSONNALISATION D'UN FORMULAIRE

Sélectionner le formulaire et choisir Modifier, en cliquant sur l'objet il est sélectionnée et vous pouvez le déplacer et modifier sa taille. Chaque élément a des propriétés et peut réagir à des événements ; propriétés et événements dépendent du type de l'objet.

Ici une liste de quelques propriétés de la zone de texte :

propriété	note
nom (name)	
visible (visibile)	
active (enabled)	se vrai permet d'insérer et modifier le texte
verruillé (loked)	se vrai permet ne permet pas de modifier le texte
hauteur, largeur, gauche, haut (height, width, left, top)	dimension et position sur la fenêtre
police, taille, type, … (fontname, fontsize, …)	
Source contrôle (controlsource)	peut être un champ d'une table ou une requête ou une expression

Table 2-4 propriétés de la zone de texte

Automatiquement le formulaire est engendré avec des boutons standard pour le déplacement entre les données.

2.2.4.1 Dimension des fenêtres

Les dimensions de hauteur et largeur, dans les fenêtres pas à l'écran plein, ils devraient être dans la proportion 1:1.6, ça correspond à *le nombre d'or*[7] à peu près 1.61803… qui vient de la proportion 1 : x = x : 1 – x.

2.2.4.2 Alignement et taille des objets

- **Aligner des objets sur une fenêtre** il faut les sélectionner : click de la souris en poussant la touche shift, un click sur la guide numéroté en haute ou à gauche sélectionne tous les objets en vertical ou horizontal. Apres Format, Aligne … .
- **Même taille** : sélectionner et Format, Taille … .
- **Petits modification de taille** : sélectionner et touche shift avec touches directionnelles.
- **Petits déplacement** : sélectionner et touche control avec touches directionnelles.

2.2.4.3 Changer un texte en liste

On peut modifier le type d'objet, par exemple on peut modifier une zone de texte en zone de liste ou dans un combo box de quelles choisir des données ; ceci est surtout utilisé où les valeurs possibles sont en nombre fixe et limité, ou doivent appartenir à une autre table, par exemple un choix de zone géographique, ou le nom du matériel qui est dans la table `Magasin`.
La procédure est :

- cliquer sur l'objet avec la touche droite de la souris,
- choisir Remplacer par ...,
- choisir Zone de liste déroulante ....

Pour indiquer les données à choisir par le combo box :

- sélectionner les propriétés du combo box,
- onglet Données,
- la propriété Origine source permet de choisir entre : Table/Requête, Liste valeurs, Liste des champs,
- dans la propriété Contenu :
  - si Table/Requête on peut choisir dans une liste de Tables et Requêtes ou créer une requête ; le résultat est une ou plusieurs colonne de données, la propriété Colonne lié indique la colonne de control ; dans onglet Format la propriété Nbre colonne indique le nombre de colonnes affichées, la propriété Largeurs colonnes donne la largeur des colonnes affichées, par exemple si on veut pour choisir le nom du matériel, les propriétés vues sont semblables à :
    - Origine source : `Table/Requête`
    - Contenu : `Magasin`
    - Colonne lié : `1` c'est le CodMateriel
    - Nbre colonnes : `3` CodMateriel, Description, Quantité
    - Largeurs colonnes : `0cm;3cm;1cm`

    L'effet est que on verra seulement Description et quantité.
  - si Liste valeurs on insère une liste de valeurs entre apostrophes et séparé par `;` (point virgule), par exemple `'F';'M'`.
  - si Liste des champs on choisit dans une liste de Tables et Requêtes, le résultat sera la liste des noms des champs.

2.2.4.4 Insertion de données pris d'une autre table

On peut prélever des informations appartenant à une autre table par les fonctions d'agrégation comme `RechDom`, `SomDom`, `CpteDom`[8], etc, à l'aide du Générateur d'expression (balise avec ... à droite de la proprieté Source contrôle) :

- ajouter les textes où on veut insérer les données d'une autre table,

[7] Le *nombre d'or*, habituellement désigné par la lettre φ(*phi*) de l'alphabet grec en l'honneur de Phidias, sculpteur et architecte grec du Parthénon.
[8] La syntaxe du Générateur d'expression diffère de celle du Basic : les nom de fonctions dépendent de la langue et les paramètres sont séparé par `;`.

- dans l'onglet Données insérer dans la propriété Source contrôle l'instructions `DLookup`, avec la syntaxe : `= RechDom ("nomchamp";"nomtable";"condition")`,
- éventuellement protéger le champ (propriétés Activé et Verruillé).

condition ha la même structure de la condition de la clause `WHERE` dans les commands `SELECT` SQL.

Exemples :

- `=RechDom("[Compound]";"Farm";"[Nom]='" & [Nom] & "'")`
- `=SomDom("[Qty]";"Med";"[Nom]='" & [Nom] & "'")`

2.2.5 FORMULAIRE ET SUB FORMULAIRE

Pour avoir un formulaire pour insérer un autre formulaire lié:

- Ouvrir en modification le formulaire conteneur,
- Choisir de la boite à outils la balise Sous-formulaire/Sous-Etats  ,
- Choisissez sur le formulaire où insérer le sous-formulaire ; un Assistant conduit, et les noms des champs de liaisons sont égaux, il y a presque rien à ajouter.

2.3 REQUETES

Les requêtes sont utilisées pour lier ensemble les données de deux ou plusieurs tables ou requêtes, par exemple la requête pour obtenir une facture utilisera les données des tables *Ventes*, *Clients* et *Magasin*. Cette technique exige que ces tables aient des champs en commune, voir le schéma de Figure 7 le champ *CodeClient* met en relation les tables Clients et Ventes et le champ *CodeMatériel* lie les tables Ventes et Matériels.

Pour engendrer une requête on peut choisir entre Créer une requête en mode Création o Créer une requête à l'aide de l'Assistant.. Dans le premier cas on choix les tables et/ou les requêtes voulues. Si les champs de liaison ont le même nom dans les tables (comme dans l'exemple), ACCESS effectue lui-même la liaison, dans le cas contraire ont doit indiquer les champs de liaisons (bouton droit de la souri pressé sur le nom de la table que on veut lier, joindre le nom de l'autre table et relaisser la touche).

Il faut indiquer les champs insérés dans la requête : bouton droit de la souri pressé sur le nom de la table que on veut insérer et le déplacer sur une cellule en bas.
Dans la cellule en correspondance a Tri : et au nom du champ on peut indiquer l'éventuel tri de la requête.
On peut insérer des critères de choix Dans la cellule en correspondance a Critères : et au nom du champ, par exemple si on indique `"F"` en correspondance du champ Sexe, la requête affichera seulement les ligne des femelles.

2.4 ETATS

Pour engendrer une requête on peut choisir entre Créer un état en mode Création o Créer un état à l'aide de l'Assistant. Ils sont des façon de travaille analogues à celle des formulaires.
Comme les formulaires les états ont propriété et événements, en particulier l'événement Sur ouverture peut être utilise pour changer le triage et/ou choisir entre les données.

2.4.1 CREATION ET MODIFICATION D'UN ETAT

Les Etats se préparent et modifient presque comme les formulaires.

2.4.2 AJOUTER DES TOTAUX

Il est possible ajouter des totaux ou des calcules à un état dans les zones En-tête de groupe et Pied de groupe pour les calcules partiels, et dans la zone Pied d'état pour les calcules globaux. Ces formules sont dans une zone de texte ou la propriété `ControlSource`, exemple `=Somme([Montant])`.

2.4.3 CHOISIR LES DONNEES

Dans l'événement `Sur ouverture` d'un état on peut insérer des choix sur les données en modifiante la propriété Filtre de l'état, mais aussi le command `DoCmd.OpenReport`, qui peut être engendré automatiquement par le bouton de création d'état, prévoit un paramètre utilisé pour choisir entre les fiches, dans l'exemple il y a les instructions *VBA* pour choisir les données (en **gras** les instructions ajouté ou modifiées).

```
Private Sub cmdEtat_Click()
On Error GoTo Err_cmdEtat_Click
    Dim stDocName As String
    where = "year(DatePrete) = " & Annee & " AND Month(DatePrete) = " & Mois
    stDocName = "etLouages"
    DoCmd.OpenReport stDocName, acPreview, , where
Exit_cmdEtat_Click:
    Exit Sub
Err_cmdEtat_Click:
    MsgBox Err.Description
    Resume Exit_cmdEtat_Click
End Sub
```

Figura 9 Exemple de état avec choix des données

2.5 QUELQUE FONCTION DE VBA

2.5.1 CONTROLES PERSONALISÉS

Il est possible associer à un événement des élaborations en Visual Basics. En sélectionnant, dans les propriétés, l'événement, paraît à droite un bouton, qu'il propose une liste de laquelle faut choisir Générateur de code, tel choix crée en Visual Basics le squelette d'une procédure. Ici de suite un exemple de code activé quand on sort du champ de nom `tst` et qui transforme le texte en minuscule sauf la première lettre.

```
tst = UCase(Left(tst, 1)) & LCase(Mid(tst, 2))
```

2.5.2 CONTROLES SUR UN CHAMPS

Dans les événement Sur sortie (OnExit) et Sur perte focus (OnLostFocus) on peut contrôler ou modifier le donné inséré ; la différence entre les deux événement est que dans Sur sortie on peut indiquer de rester avec le curseur sur la zone de texte (voir exemple).

```
Private Sub Sexe_Exit(Cancel As Integer)
Sexe.Value = UCase(Sexe.Value)
If InStr("MF", Sexe.Value) = 0 Then
    MsgBox "Sexe interdit!"
    Cancel = True
End If
End Sub
```

2.5.3 OPERATIONS PRE-AFFICHAGE DE DONNEES

Dans les événement du formulaire Sur Affichage (OnCurrent) on peut effectuer des opérations ; dans l'exemple pour les lignes affiché on visualise les valeurs de prix et Stock prélevé de la table Magasin.

```
Private Sub Form_Current()
Prix = 0
Stock = 0
If Not Me.NewRecord Then
    Prix = DLookup("Prix", "Magasin", "CodeMateriel =" & CodeM)
    Stock = DLookup("Stock", "Magasin", "CodeMateriel =" & CodeM)
End If
End Sub
```

2.5.4 EXAMEN DES OBJET D'UN FORMULAIRE

Il est possible accéder à les objet d'un formulaire, l'exemple change le contenu des étiquette (pour changer la langue).

```
Public Sub ImpostaLingua(Frm)
' internazionalizzazione delle Forms presuppone label in Italiano
If Not IsEmpty(Lingua) And Lingua <> "Italiano" Then
     For Each MyCtrl In Frm.Controls    ' Esegue un'iterazione in ogni elemento
       If MyCtrl.ControlType = acLabel Then
         Label = Nz(DLookup(Lingua, "Dizionario", "Italiano = '" &  _
                 MyCtrl.Caption & "'"))
```

```
          If Label <> "" Then MyCtrl.Caption = Label
        End If
      Next
End If
End Sub
...
ImpostaLingua Me
...
```

2.6 LIGNE GUIDE POUR BATIR UNE APPLICATION

Ce paragraphe indique des techniques pour bâtir une application, en particulier seront traité :

- conseil sur le nom des objets,
- accéder aux différentes fonctionnalités de l'application (panneau de contrôle),
- rendre l'application utilisable par différents subjectifs (personnalisation),
- séparer les données de l'application,
- utiliser des sources de données préexistantes,
- protéger et contrôler les données

2.6.1 CONSEIL SUR LE NOM DES OBJETS

On peut utiliser quelconque nom pour les objets, toutefois il est fort conseillé d'utiliser toujours des nom significatifs, surtout pour les objet auxquels on ajute le traitement des événements. La figure qui suive utilise, partiellement, la nomenclature Hongrois : des lettres minuscules initiales qui individus l'objet, suivi par le nom de l'objet dont la premier caractère est majuscule :

Nom Objet	Type	Note
`Ventes`	Table	Remarquez le pluriel
frmVentes	Formulaire	Formulaire relatif aux ventes
rqtVentesClients	Requête	Requête relatif aux ventes des Clients
eVentesClients	Etat	Etat relatif aux ventes des Clients
cmdVentes	Bouton	Bouton qui démarre le formulaire ou l'état des Ventes
etqNouveau	Etiquette	

Vous pouvez faire exception pour le Tables et les nom des Zone de texte, surtout car ACCESS donne à les Zone de texte relatives à champs de table et requêtes, le nom du champ même, par exemple si la table Clients contient le champ `Adresse`, sur le formulaire obtenu par l'Assistant, l'objet zone de texte correspondant aura le nom `Adresse`.

Il est important utiliser le même nom (et même taille de donnée) pour les champ utilisé pour lier ensemble les table, par exemple si la table `Ventes` contient `CodeClient` et `CodeMateriel` (format numérique Entier long), et les tables `Clients` et `Magasin` contiennent respectivement `CodeClient` et `CodeMateriel`, ACCESS sera capable de faire les liaisons automatiquement pour vous.

2.6.2 LE FORMULAIRE DE CONTROLE

Le "formulaire de contrôle" est un formulaire qui contient les boutons pour démarrer les principales fonctions de l'application, comme par exemple :

- accéder aux formulaires de gestion des données,
- production d'états,
- fonction de recherche et d'extraction des données,
- élaborations techniques comme le sauvetage de la base des données et la modification des personnalisations,
- protection et discrétion,
- terminer l'application.

Le formulaire doit être créé avec la fonction Créer un formulaire en mode Création.

Du panneau des instruments sélectionner les boutons et avec la souris portez-le sur le formulaire. Pour les principales opérations ACCESS construit automatiquement le code Visual Basic.

Pour qu'ACCESS au commencement présente la " fenêtre de commande", il faut accéder à la voix de menu Outils, choisir Démarrage.... et choisir le nom du panneau de démarrage du combo box Afficher formulaire/page.

2.6.3 PERSONALISATION

Une application peut être utilisé par différents utilisateur, dans ce cas il est important déterminer les informations variables, en général sont des informations comme la nom de la Société, l'adresse etc. Ces données doivent être mémorisé dans une table.

2.6.4 PROTECTION ET SURETE

2.6.4.1 Protection

L'aide en ligne de ACCESS explique plusieurs techniques pour la protection.

Pour permettre l'utilisation de l'application seulement aux personnes autorisées on peut :

- insérer la requête de nom d'utilisateur et de mot de pas, ceci on le doit gérer avec un formulaire et une table qui contienne les information de l'utilisateur. Cette technique est faible parce que on peut voir le contenu des tables.
- Utiliser les protection de Windows, qui sont accessibles par le menu Outils et dont la plus simple est l'utilisation d'un mot de pas.

Pour protéger l'application de modifications accidentelles ou voulues est possible de bloquer formulaires et code Visual Basic :

- menu Outils,
- choisir Utilitaires de base de données,
- choisir Créer un fichier MDE...,
- confirmer ou changer le nom proposé,
- cliquez sur le bouton Enregistrer.

Cette opération engendre l'application sous forme de fichier `.mde`. qui sera distribué à l'utilisateur final au lieu de l'application `.mdb`.

2.6.4.2 Sûreté

Au-delà à la protection de l'application d'accès pas autorisés, il faut prévoir le sauvetage périodique des données, par exemple à chaque fin de l'application et/ou à chaque semaine. Le sauvetage devrait être fait sur une unité de mémorisation différente de celle de travail (autre disque dur, pen drive, CD, etc.). Il faut prévoir dans la table de personnalisation la mémorisation de la date du dernier sauvetage et le control au démarre de l'application.

Le sauvetage sur le même disque est tolérable seulement pour les sauvetages en fin de travail.

Ici à droite **Figure 10** une procédure de sauvetage sous forme de procédure DOS `.bat`, il est prévu trois générations de la même Base de Données, pour exécuter le sauvetage ajouter une instruction VBA comme la suivante :

```
Call Shell("D:\CONDOR\SERMIG\FARO\sauve.bat", vbNormalFocus)
```

```
echo off
REM où sont les données
D:
cd D:\CONDOR\SERMIG\FARO
if exist faro3.mdb del faro3.mdb
if exist faro2.mdb ren faro2.mdb faro3.mdb
if exist faro1.mdb ren faro1.mdb faro2.mdb
copy faro.mdb faro1.mdb
pause Presser une touche pour continuer
exit
```

Figure 10

2.6.5 SEPARATION ENTRE DONNEES ET APPLICATION

Presque toutes applications ont besoin de modification : pour améliorer, pour introduire des nouvelles fonctions ou modifier et enfin aussi pou corriger des fautes.

Avec un Base de Données qui contient données et application, modifier celle-ci peut provoquer des dommages aux données. Pour éviter ceci ACCESS permet de séparer les deux composants, c'est-à-dire que

l'application sera composé par deux (ou plus) Base de Données ACCESS, l'une avec les données et l'autre avec l'application. Cette dernière au lieu des tables aura des liaisons aux tables présentes dans l'autre.
La procédure pour obtenir la liaison est la suivante :

- ouvrir ACCESS applications et dans la gestion des Table
- choisir Nouveau 
- Importer la table,
- chercher le fichier ACCESS avec les données,
- choisir toutes les tables dont on veut la liaison,
- cliquer sur le bouton OK.

2.6.6 UTILISER DES SOURCES DE DONNEES PREEXISTANTES

2.6.6.1 De EXCEL à ACCESS

Une feuille D'EXCEl qui contienne des données structuré peut être transformé aisément en une table d'ACCCES, pourvu que la première ligne contienne les entêtes des colonnes (qui seront transformé en nom de champ) :

- choisir Fichier,
- Données externes,
- Importer …,
- Dans Type de fichier : choisissez Microsoft Excel (*.xls)
- Choisissez le fichier EXCEL et pousser le bouton Importer
- Apparaître l'Assistant Importation de feuille de calcul qui vous permet d'importer les données.

Exemple :

Nom	Adresse	Ville	Dette	DMaJ
Ector Coulibaly	Rue Bamako	Kati	50000	08/10/2006
Jean Ross	Rue Oriani	Turin	150000	03/10/2006
Amina Toure	Rue Bamako	Bamako	25000	08/09/2006
Fatuma Diallo	Rue 99 Koko	Kati	50000	08/10/2006