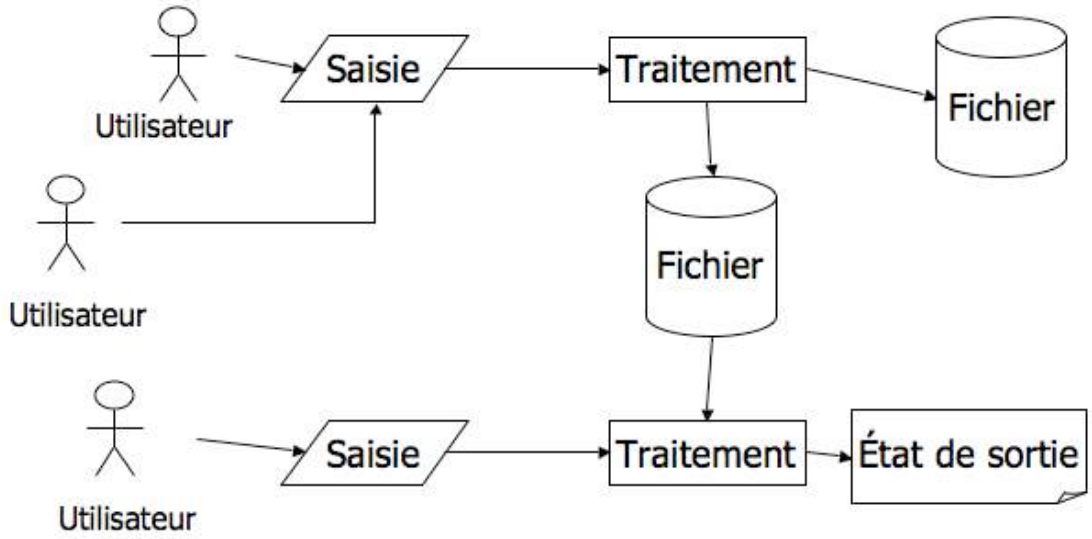
Systèmes de Gestion de Bases de Données

1. Introduction

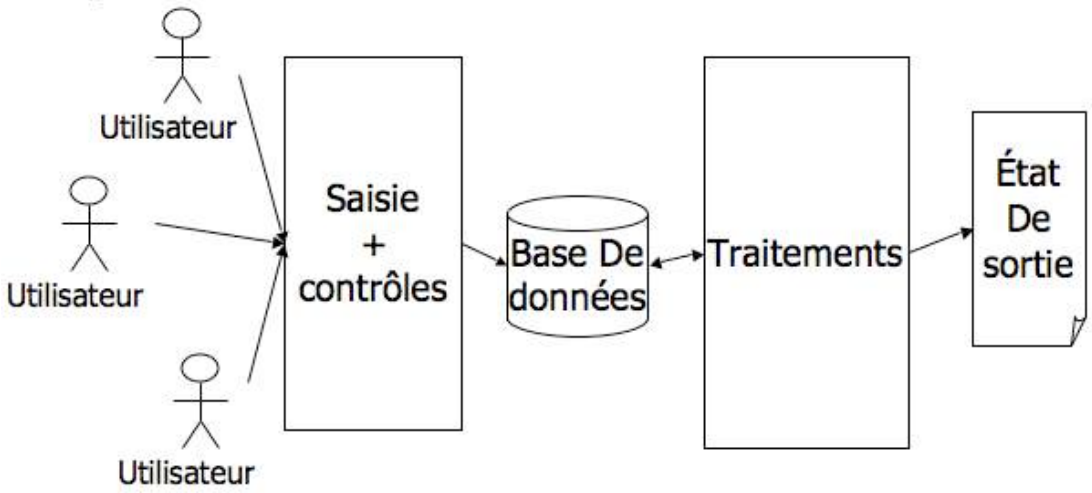
1.1. Organisation en fichiers :



1.2. Limites de l'organisation en fichiers :

- Particularisation de la saisie et des traitements en fonction des fichiers ⇒ un ou plusieurs programmes par fichier
- Contrôle différé des données ⇒ augmentation des délais et du risque d'erreurs
- Particularisation des fichiers en fonction des traitements ⇒ grande redondance des données

1.3. Organisation en base de données :



1.4. Organisation en base de données :

- Uniformisation de la saisie et standardisation des traitements (ex: tous les résultats de consultation sous forme de liste et de tableaux)
- Contrôle immédiat de la validité des données
- Partage de données entre plusieurs traitements ⇒ limitation de la redondance des données

1.5. Définitions

- **Base De Données (BDD)** : ensemble structuré de données enregistrées avec un minimum de redondance
- **Système de Gestion de Base de Données (SGBD)** :Logiciel(s) assurant structuration, stockage, maintenance, mise à jour, et consultation des des données d'un BDD

1.6. Couches SGBD

- Le système de gestion de fichiers
  - Gestion du stockage physique de l'information
- Le SGBD interne
  - Placement et assemblage des données, gestion des liens et de l'accès rapide
- Le SGBD externe
  - Présentation et manipulation des données aux concepteurs et utilisateurs, gestion de langages de requêtes élaborés

1.7. Objectifs des SGBD

- **Indépendance physique** : un remaniement de l'organisation physique des données n'entraîne pas de modifications dans les programmes d'applications
- **Indépendance logique** : un remaniement de l'organisation logique des fichiers n'entraîne pas de modifications dans les programmes d'applications
- **Manipulation facile des données** : un utilisateur non informaticien doit pouvoir manipuler simplement le données
- **Administration facile des données** : un SGBD doit fournir des outils pour décrire les données, permettre le suivi de ces structures et autoriser leur évolution (administrateur de BDD)
- **Efficacité des accès aux données** : garantie d'un bon *débit* (nb. de transactions /s) et d'un bon *temps de réponse* (temps moyen d'attente pour une transaction)
- **Non redondance des données** : éviter la duplication d'informations pour diminuer le volume de stockage et éviter les incohérences

■**Cohérence des données** : ex: l'âge d'une personne est un nombre positif. Le SGBD veille à ce que les applications respectent cette règle (*contrainte d'intégrité*)

■**Partage des données** : utilisation simultanée des données par différentes applications

■**Sécurité des données** : les données doivent être protégées contre les accès non autorisés ou en cas de panne

1.8. Différents types de BD

■Bases hiérarchiques (schéma arborescent)
■Bases réseaux (id. mais plus rapides) 20%
■Bases relationnelles (tables et algèbre relationnel) 75%
■Bases déductives (tables et logique du 1er ordre)
■Bases objets (formalisme objet d'héritage)

1.9. Niveaux de description

■Niveau interne : structure de stockage des données : type de fichiers, accès aux données

■Niveau conceptuel : description abstraite de la réalité du domaine d'application

■Niveau externe : ensemble des vues (façon dont les données sont perçues) qu'ont les groupes d'utilisateurs

1.10. Fonctions des SGBD

■Description des données : *Langage de Description des données* (LDD)

■Recherche des données (langage de manipulation de données LMD)

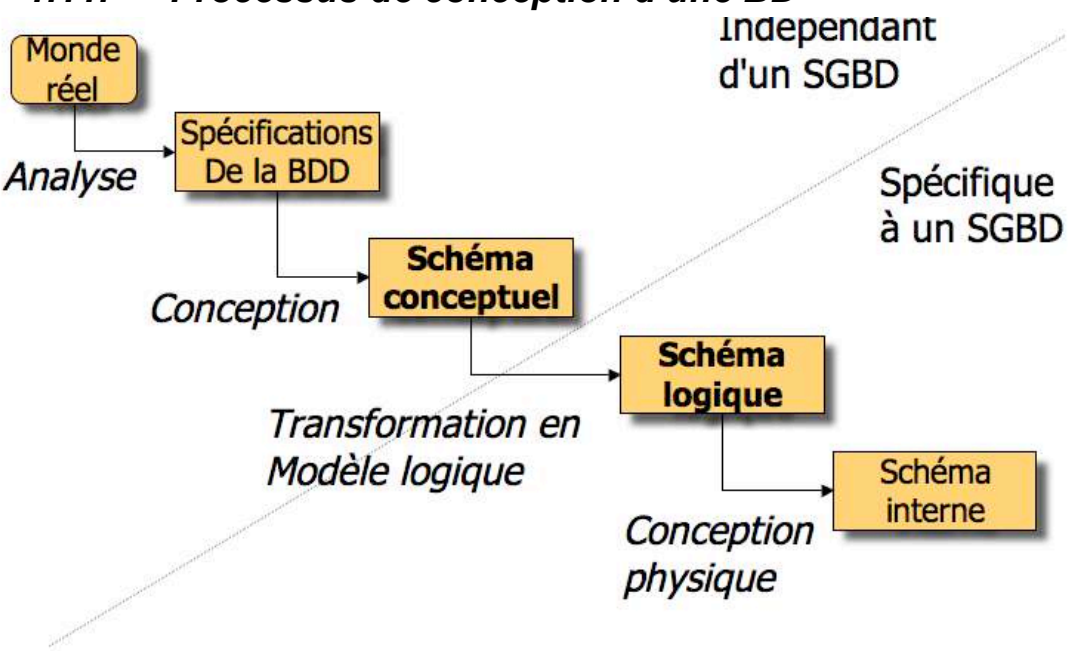
■Mise à jour des données (langage de manipulation de données LMD)

■Transformation des données (langage de manipulation de données LMD)

■Contrôle de l'intégrité des données

■Gestion de transactions et sécurité

1.11. Processus de conception d'une BD



2. Le modèle E/A

2.1. Analyse et conception de BD

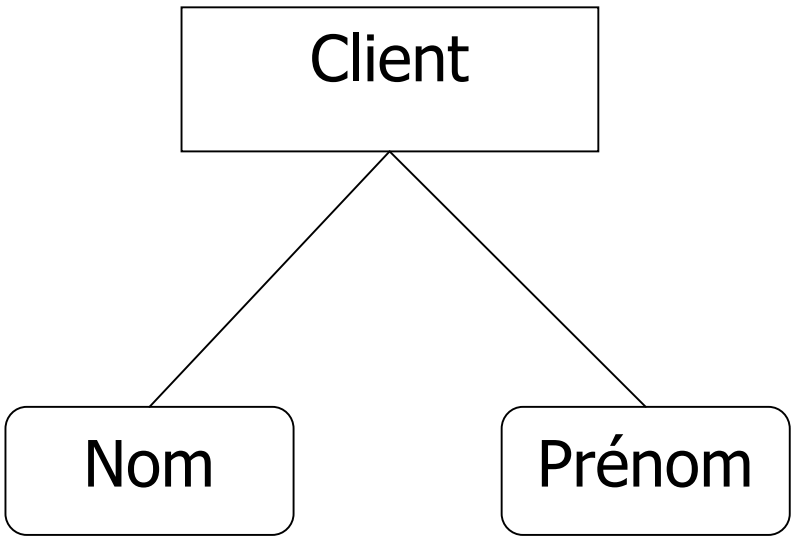
- La phase de conception nécessite des choix qui ont d'importantes répercutions
- Exprimer les besoins avant de l'implanter : l'analyse
- L'analyse présente de manière abstraite le travail à réaliser
- On utilise un MCD (Modèle Conceptuel des Données, méthode MERISE)

2.2. Le formalisme E/A

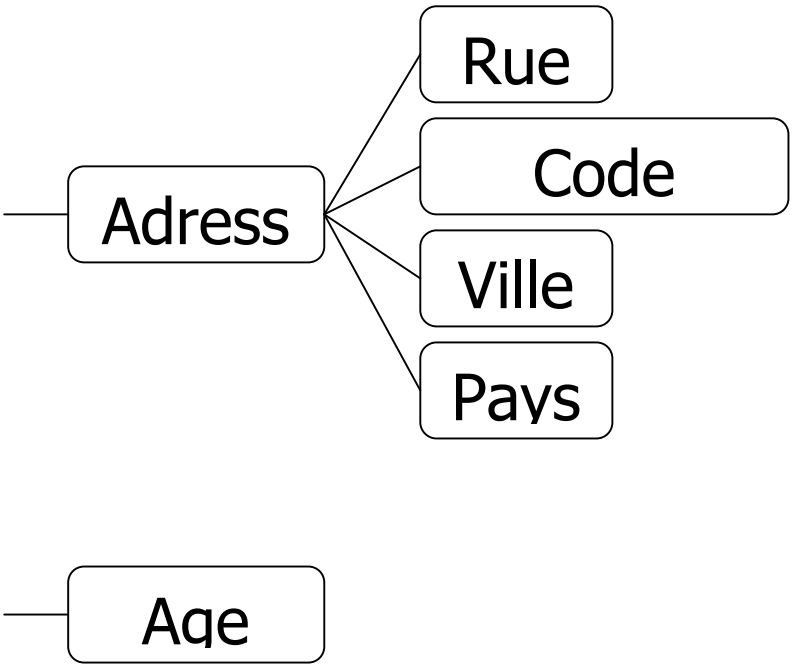
- E/A signifie entité/Association
- Modèle conceptuel des années 1970
- Il utilise une représentation graphique
- Le modèle E/A est un outil permettant d'analyser les situations du monde réel

Entités et attributs

- **Entité** : objet (abstrait ou concret) qui peut être distingué d'un autre objet
- **Classe d'entités** : ensemble d'entités similaires
- **Attribut** : propriété caractéristique d'une entité



- Attribut *composé* : subdivisé en attributs simples (ex: Adresse)
- Attribut *dérivé* : dont le valeur est calculée (ex: Age d'après la date de naissance)



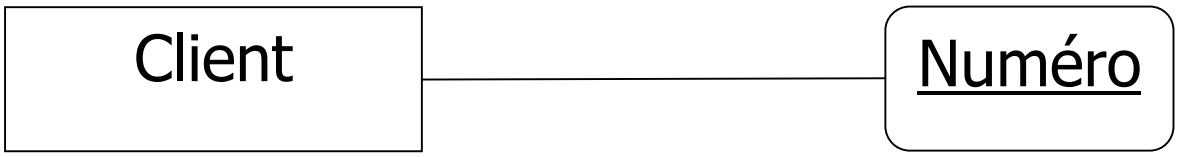
■**Domaine** : ensemble des valeurs que peut prendre un attribut (ex: Domaine d'un PRIX est l'ensemble est réels positifs)

■**Occurrence** : valeur particulière d'un attribut ou d'une entité (ex: Bleu, Rouge sont des occurrences d'un attribut Couleur)

Identifiant et clé

■Une clé ou un identifiant est un attribut ou un ensemble d'attributs dont les valeurs identifient de façon unique les occurrences d'un type d'entité

■Ex: le numéro de sécurité sociale est une clé

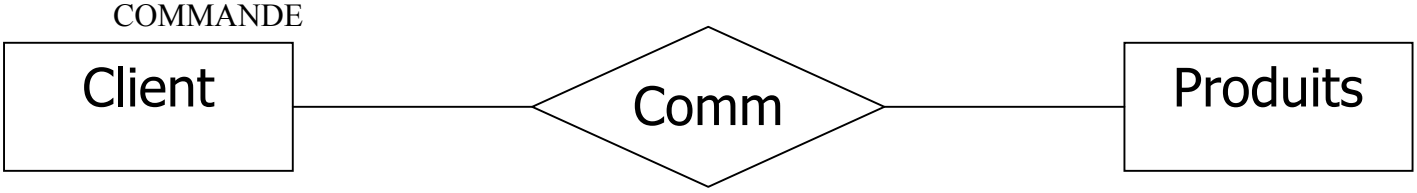


2.3. Association

■Association : regroupement d'entités traduisant une réalité c-a-d une liaison entre des entités

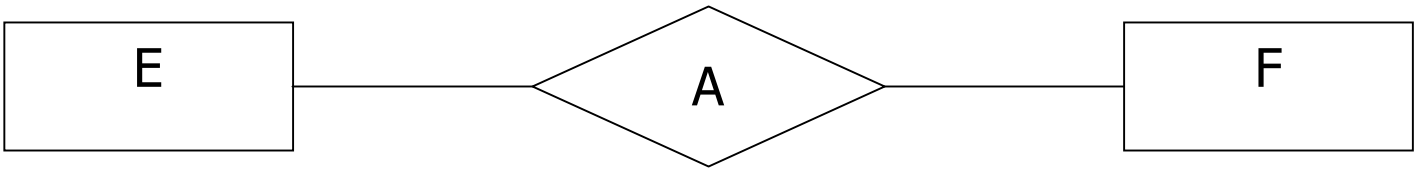
■Les associations similaires sont regroupées en classes d'associations

■Les entités CLIENT et PRODUITS sont dites participantes à la relation COMMANDE



2.4. Types d'associations

- Le type d'association caractérise le nombre de liens entre entités
- Associations N-aires entre N entités
- Associations binaires entre deux entités
  - Association 1:1 (un à un)
  - Association 1:n ou 0:N (un ou zéro à plusieurs)
  - Association n:m (plusieurs à plusieurs)

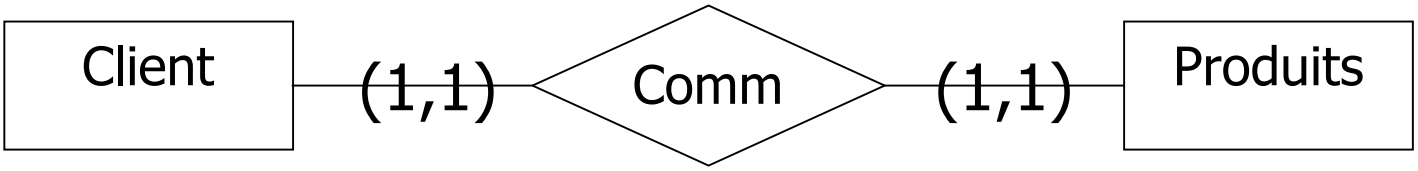


2.5. Cardinalité

- La cardinalité d'un couple entité-association (E,A) est la données d'un couple d'entiers (m,M)
- Elle est définie par une cardinalité minimum m et maximum M
- m définit le nombre minimum d'associations de classe A pouvant exister pour une entité de classe E

2.6. Association 1 : 1

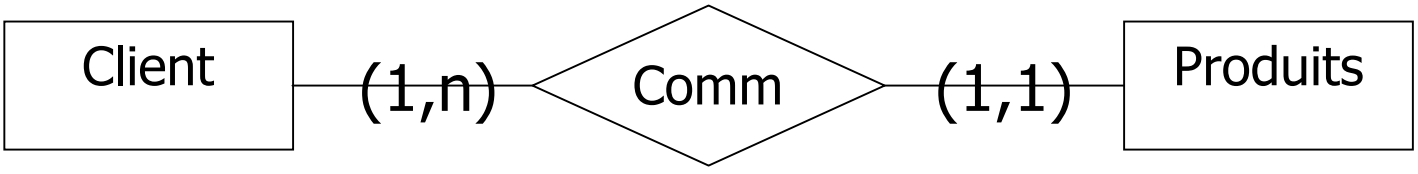
Ex: un client donné ne commande qu'un seul produit. Un produit donné n'est commandé que par un seul client.



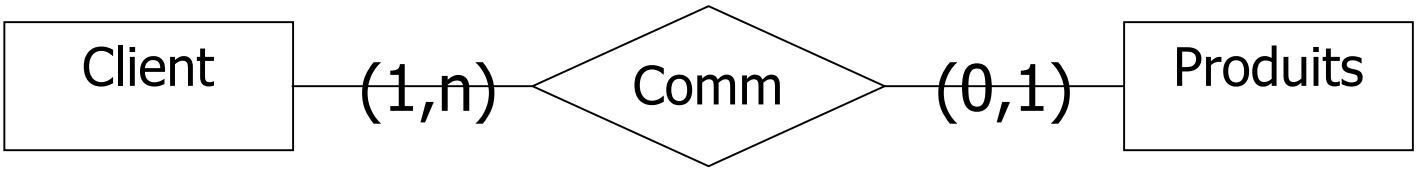
- (m,M) m : cardinalité minimum, M : cardinalité Maximum
- Un client commande de m à M produits

2.7. Association 1 : n

- Ex: un client donné commande plusieurs produits. Un produit donné n'est commandé que par un seul client.

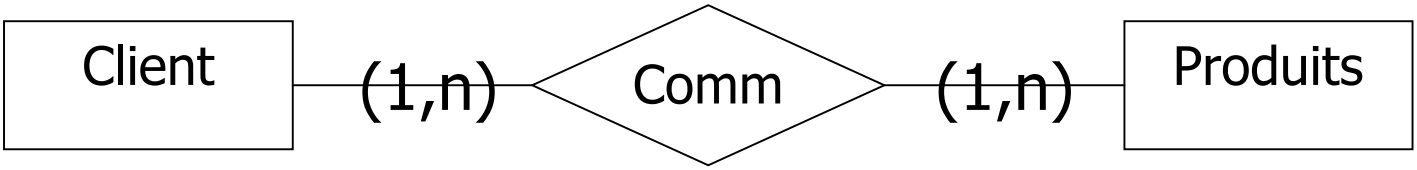


■Ex: un client donné commande plusieurs produits. Un produit donné n'est commandé que par un seul client mais peut ne pas être commandé



2.8. Association m : n

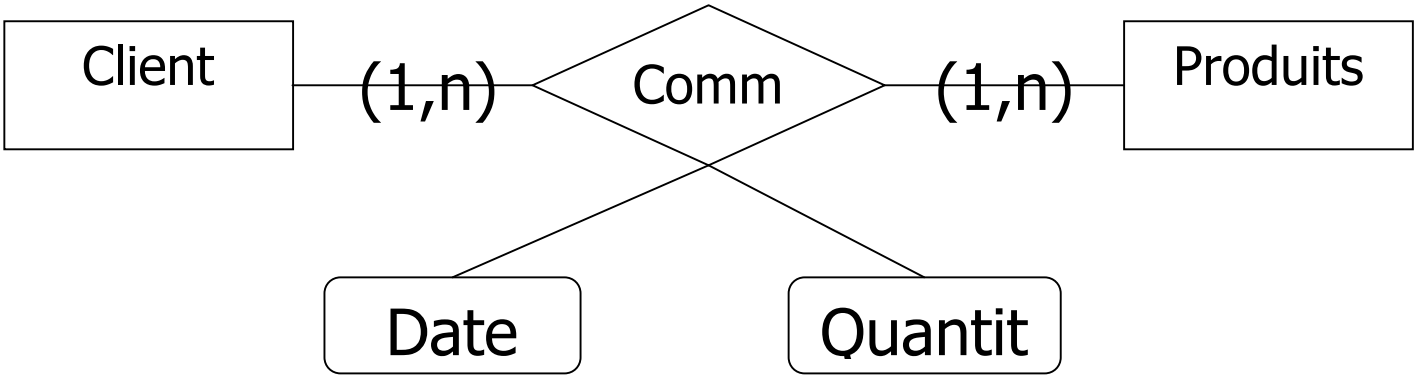
Ex: un client donné commande plusieurs produits. Un produit donné est commandé par plusieurs clients.



Les cardinalités "un à plusieurs" (1,N) peuvent être aussi "zéro à plusieurs" (0,N)

2.9. Attributs d'association

Attribut d'association : dans une association m:n, il est possible de caractériser l'association par des attributs
Ex: une commande passée à une date donnée et qui concerne une certaine quantité

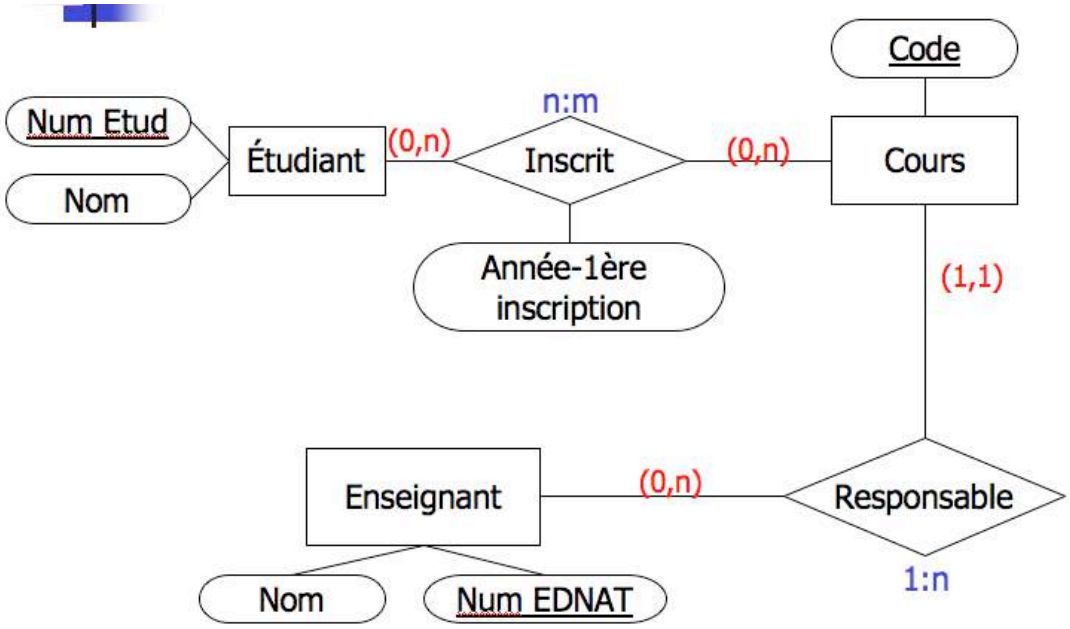


2.10. Analyse et conception

■Marche à suivre pour produire un schéma E/A

1. Identifier les entités
2. Identifier les associations entre entités
3. Identifier les attributs de chaque entité et de chaque association
4. Évaluer les cardinalités des associations
5. Identifier le type d'association

2.11. Exemple de diagramme E/A



2.12. TD numéro 1 et 2 Le modèle E/A

Exercice 1

Nous souhaitons concevoir une base de données pour une banque incluant des informations sur les comptes et les clients. L'information relative à un client consiste en son adresse, son téléphone ainsi que son numéro de sécurité sociale. Les comptes ont un numéro, un solde, un type (courant, épargne, etc.). Il faut en outre pouvoir déterminer quels sont les comptes d'un client donné ainsi que le client propriétaire d'un compte. On respectera les contraintes suivantes : pour chaque compte, il n'y a qu'un seul titulaire, un client ne peut posséder qu'un seul compte, les adresses sont composées d'un numéro, d'un nom de rue, d'une ville et d'un code postal. Un client peut avoir plusieurs adresses. Un client peut avoir plusieurs numéros de téléphone pour chaque adresse. Ceci implique de conserver pour chaque adresse quels numéros de téléphone lui sont associés. Donner une représentation entités-associations pour la gestion de cette base de données.

Exercice2

Donner une représentation entités-associations pour la gestion d'une base de données stockant de l'information relative à des équipes, des joueurs et leurs fans. On suppose que chaque équipe a un nom, un capitaine et une couleur de maillot, chaque joueur a un nom, chaque fan a un nom, des équipes favorites, des joueurs favoris et une couleur préférée.

Exercice 3

Donner une représentation entités-associations pour la gestion d'une base de données stockant de l'information relative à une bibliothèque contenant des ouvrages pouvant être empruntés. Un ouvrage est caractérisé par un numéro identifiant, un titre, un auteur et un éditeur. En outre on décrit un ouvrage par un certain nombre de mots clés numérotés qui indiquent les sujets qui y sont traités. La bibliothèque dispose d'un ou de plusieurs exemplaires de chaque ouvrage. L'exemplaire, qui est en quelque sorte la matérialisation d'un ouvrage, est identifié par un numéro et caractérisé par sa position dans les rayonnages et sa date d'achat. Un exemplaire peut être emprunté par un emprunteur, qui peut en emprunter plusieurs. Un emprunteur est identifié par un numéro et caractérisé par son nom et son adresse.

Exercice 4

Donner une représentation entités-associations pour la gestion d'une base de données stockant de l'information relative à la gestion des vols d'une compagnie aérienne. Un vol est un parcours aérien caractérisé par un numéro, une ville de départ, une ville d'arrivée, une distance, une fréquence. Lorsqu'un vol est programmé pour une date déterminée, il constitue un départ. Un vol n'est programmé qu'une seule fois dans une journée à l'heure prévue. Un certain nombre de passagers peuvent être enregistrés pour un départ. Un passager est caractérisé par son nom, son adresse et son numéro de téléphone. Un avion est affecté à chaque départ. Un avion est caractérisé par un numéro, un type et une capacité. Un avion utilise une certaine quantité de carburant pour accomplir le trajet. Cette dernière dépend des conditions atmosphériques et donc de la date. Un certain nombre de personnels sont affectés à chaque départ. Un

membre du personnel est caractérisé par son nom, son adresse et son numéro de téléphone.

Exercice 5

Les animaux d'un zoo suivent chacun un régime alimentaire. Un régime est constitué d'un mélange d'ingrédients, chacun en quantité déterminée. Le régime d'un animal peut varier d'un jour à l'autre. Chaque animal est caractérisé, en fonction de son espèce, par ses besoins minima et maxima en nutriments (calcium, protéines, etc.) exprimés en mg par unités de poids de l'animal. Ces besoins sont donc fonction de l'espèce de l'animal. On connaît la teneur de chaque ingrédient en nutriments, exprimée en mg par kg d'ingrédient. Chaque ingrédient a un coût unitaire. Chaque animal requiert des soins qui sont évalués en francs par jour. Ces soins peuvent varier d'un jour à l'autre.

3. Le modèle relationnel

3.1. Généralités

Le modèle relationnel est un *modèle logique* associé aux SGBD relationnels (ex: Oracle, Access, Paradox, dBase, SQLServer)

- Objectifs du modèle relationnel :
  - Indépendance physique
  - Traitement du problème de la redondance
  - Propose une démarche pour :
    - La description (LDD)
    - L'interrogation (LMD)

3.2. Notions de base

- Une BDD relationnelle est un ensemble de schémas de relations et dont les occurrences sont des n-uplets de ces relations
  - Table = relation
  - Noms des colonnes = attributs
  - Ligne = n-uplet

Schéma de base = l'ensemble des schémas de relations

Bières	Nom	Fabriquant
	1664	Kronenbourg
	Mort subite	Jeanlain
	...	...

3.3. Relations, attributs

- Une *Relation* est un ensemble d'attributs $\{A_1,A_2,\ldots,A_n\}$.

Ex: la relation PRODUIT est l'ensemble des attributs {NumProd, Dési, PrixUni}

- Chaque attribut A_i prend ses valeurs dans un *domaine* $dom(A_i)$

Ex: PrixUni ∈]0,10000]

3.4. N-uplets

Un n-uplet est un ensemble de valeurs
$t=<V_1,V_2,\ldots,V_n>$ où $V_i \in dom(A_i)$ ou bien V_i est la valeur nulle (NULL)

Ex: <112,'raquette de tennis',300> est un n-uplet de la relation PRODUIT
Notation : R(A_1,A_2,…,A_n)
Ex: PRODUIT(NumProd,Dési,PrixUni)

3.5. Contraintes d'intégrité

■**Clé primaire** : Ensemble d'attributs dont les valeurs permettent de distinguer les n-uplets les uns des autres (*identifiant*)
Ex: *NumProd* clé primaire de la relation PRODUIT

■**Clé étrangère** : Attribut qui est clé primaire d'une autre relation
Ex: connaître le fournisseur de chaque produit ajout de l'attribut *NumFour* à la relation PRODUIT

■**Notations :** clés primaires soulignées, clés étrangères *en italiques*
Ex: PRODUIT(NumProd,Dési,PrixUni,*NumFour*)

■**Contraintes de domaine** : les attributs doivent respecter un condition logique
Ex: PrixUni>0 ET PrixUni ≤ 10000

3.6. Pourquoi des relations

■Modèle très simple

■Modélisation mathématique

■Bonne adéquation avec la façon que nous avons de percevoir les données

■Modèle théorique sous-jacent à SQL qui est le plus important langage pour BDD à ce jour

3.7. Traduction Relationnel – E/A

1) Chaque entité devient une relation (une table relationnelle) dont les attributs sont:
- ■Tous les attributs de l'entité
- ■L'identifiant de l'entité qui forme la clé primaire de la relation

Ex: CLIENT(NumCli,Nom,Prénom,DateNaiss,Rue,CP,Ville)

2) Chaque association 1:1 se traduit en incluant la clé primaire d'une des relations comme clé étrangère dans l'autre relation
Ex: Si un client peut posséder un compte, on aura :
COMPTE(NumCom, Solde)
CLIENT(NumCli,Nom,Prénom,DateNaiss,Rue,CP,Ville, *NumCom*)

3) Chaque association 1:n se traduit en incluant la clé primaire de la relation dont la cardinalité maximale est n comme clé étrangère dans l'autre relation
Ex:
PRODUIT(NumPord,Dési,PrixUni,*NumFour*)

FOURNISSEUR(NumFour,RaisonSoc)

4) Chaque association m:n se traduit en créant une nouvelle relation dont la clé primaire est la concaténation des clés primaires des relations participantes. Les attributs de l'association sont insérés dans cette nouvelle relation
Ex: COMMANDE(NumCli, NumProd, Date, Quantité)

3.8. Traduction E/A – Modèle relationnel : exemple complet

Schéma relationnel complet de l'exemple VPC

CLIENT(NumCli,Nom,Prénom,DateNaiss,Rue,CP,Ville)
PRODUIT(NumPord,Dési,PrixUni,*NumFour*)
FOURNISSEUR(NumFour,RaisonSoc)
COMMANDE(NumCli, NumProd, Date, Quantité)

3.9. Problème de la redondance

Soit la relation COMMANDE-PRODUIT

NumProd	Quantité	NumFour	Adresse
101	300	901	Grenoble
104	1000	902	Gières
112	78	904	Fontaine
103	250	901	Grenoble

Cette relation présente des anomalies

3.10. Anomalies détectées

- Anomalies de modification : Si l'on souhaite mettre à jour l'adresse d'un fournisseur, il faut le faire pour tous les n-uplets concernés
- Anomalies d'insertion : Pour ajouter un nouveau fournisseur, il faut obligatoirement fournir des valeurs pour *NumProd* et *Quantité*
- Anomalies de suppression : la suppression du produit 104 fait perdre toutes les informations concernant le fournisseur 902

3.11. Normalisation

La normalisation est le processus de modification relatif à la conception d'une Base de données

- Objectifs de la normalisation :
  - Suppression des problèmes de mise à jour
  - Minimisation de l'espace de stockage (élimination des redondances)
  - Sans perte d'informations

3.12. Dépendance fonctionnelle (DF)

- Soit R(X,Y,Z) une relation où X, Y et Z sont des ensemble d'attributs éventuellement vides
- **Définition:** Y *dépend fonctionnellement* de X (X→Y) si c'est toujours la même valeur de Y qui est associée à X dans la relation R
- Ex: PRODUIT(NumProd, Dési, PrixUni)
  NumProd →Dési et Dési →PrixUni

3.13. Propriétés des Dépendances fonctionnelles

- Réflexivité : Si Y ⊆ X alors X →Y
- Augmentation : si W ⊆ Z et X →Y alors X,Z →Y,W
- Transitivité : si X →Y et Y →Z
  alors X →Z

(Règles d'inférences d' Armstrong)

3.14. Propriétés complémentaires

- Pseudo-transitivité : Si X → Y et Y,Z → W alors X,Z → W
- Union : Si X → Y et X → Z alors X → Y,Z
- Décomposition : Si Z ⊆ Y et X → Y alors
  X → Z
- La notation X,Y signifie X ∪ Y

3.15. Exemple de DF

- R(Prof, Codmat, J, H, Salle)
- Un n-uplet (p,m,j,h,s) signifie « l'enseignant p enseigne la matière m dans la salle s le jour j à l'heure h »
- Contraintes :
- Une enseignant ne peut se trouver dans deux salles à la fois à un instant donné : Prof, J, H → Salle, Codmat
- Deux cours ne peuvent être donnés simultanément par la même personne à un instant donné : H, J, Salle → Prof, Codmat

La connaissance d'un enseignant implique la connaissance de la matière enseignée :
Prof → Codmat (faux si l'enseignant peut enseigner plusieurs matières)

3.16. Notions supplémentaires

Déf: La clôture F^+ d'un ensemble F est l'ensemble de toutes les DF conséquences logiques de F. On obtient la clotûre en itérant les règles d'armstrong
Déf: Un ensemble d'attributs X est une clé minimale ssi X est une clé primaire et si tout sous-ensemble de X n'en est pas une.

3.17. Formes normales

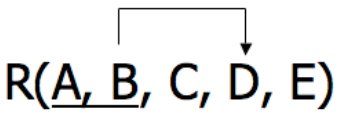
- Les schémas de table possèdent des caractéristiques particulières appelées formes en vue de la réduction des redondances
- Les relations sont classées en fonction de leurs propriétés vis-à-vis des DF à l'aide de la notion de forme normale
- La normalisation d'une BBD consiste à produire des schémas de table dans une forme normale
- Plus de degré de normalité est élevé, plus les anomalies de mise-à-jour sont réduites

3.18. Première forme normale (1FN)

- **Déf:** Une relation est en 1FN si tout attribut n'est pas décomposable
- Un attribut non décomposable est dit indivisible ou atomique
- Ex: Les relations PERSONNE(Nom, Prénom, Age) et DEPARTEMENT(Nom, Adresse, Tel) ne sont pas en 1FN si les attributs Prénom et Adresse peuvent être du type [Jean, Paul] et [Rue des Quatrans, Caen]

3.19. Deuxième forme normale (2FN)

Déf: Une relation est en 2FN si elle est en 1FN et si tout attribut non clé primaire est dépendant de la clé primaire entière et non pas d'un sous ensemble de la clé
Il faut éviter la configuration suivante :



Si D n'est pas une clé primaire, il ne faut pas que D dépende d'une partie de la clé

3.20. 2FN : Normalisation

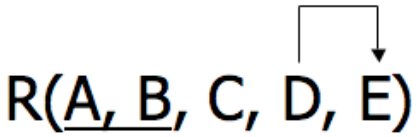
- La relation COMMANDE_PRODUIT(NumProd,NumFour, Quantité,Ville) n'est pas en 2FN car on a NumProd,NumFour → Quantité et NumFour → Ville

■La décomposition suivante donne deux relations 2FN :
COMMANDE(NumProd,NumFour,Quantité) et FOURNISSEUR(NumFour, Ville)

3.21. Troisième forme normale (3FN)

■**Déf:** Une relation est en 3FN si elle est en 2FN et qu'il n'existe aucune DF entre deux attributs non clés primaires

■Il faut éviter la configuration suivante :



■Ex: La relation COMPAGNIE(Vol,Avion,Pilote) avec les DF Vol → Avion et Avion → Pilote et Vol → Pilote est en 2FN mais pas en 3FN

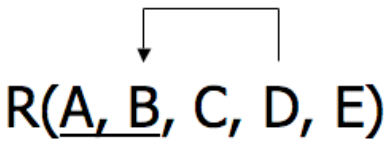
■Anomalies de mise à jour sur la relation COMPAGNIE : il n'est pas possible d'introduire un nouvel avion sur un nouveau vol sans préciser le pilote correspondant

■La décomposition suivante donne deux relations en 3FN qui permet de retrouver (par transitivité) toutes les DF :
R1(Vol,Avion) et R2(Avion,Pilote)

3.22. Forme normale de Boyce Codd (FNBC)

■**Déf:** Une relation est en FNBC si elle est en 3FN et si tout attribut ne dépend que des clés minimales du schéma

■Chaque partie gauche d'une dépendance est une clé
Il faut éviter la configuration suivante :



3.23. TD numéro 3
Passage du modèle E/A au modèle Relationnel

Exercice 1

Nous donnons ci-dessous les instances de deux relations qui pourraient constituer un fragment d'une base de données bancaire. Indiquer

a) Les attributs de chaque relation
b) Les n-uplets de chaque relation
c) Les composants d'un n-uplet pour chaque relation
d) Le schéma pour chacune des relations
e) Le schéma de la base
f) Un domaine adéquat pour chaque attribut
g) Spécifiez les clés (primaires et étrangères)

COMPTE	NUM	TYPE	SOLDE
	12345	Courant	12000
	23456	Codevi	15000
	34567	Courant	5000
	27569	PEL	7500

CLIENTS	NOM	PRENOM	NUM-CLIENT	NUM
	Dupont	Marcel	901-222	12345
	Durand	Ginette	805-333	34567
	Martin	Jérôme	555-644	27569
	Durand	Ginette	805-333	23456

Exercice 2

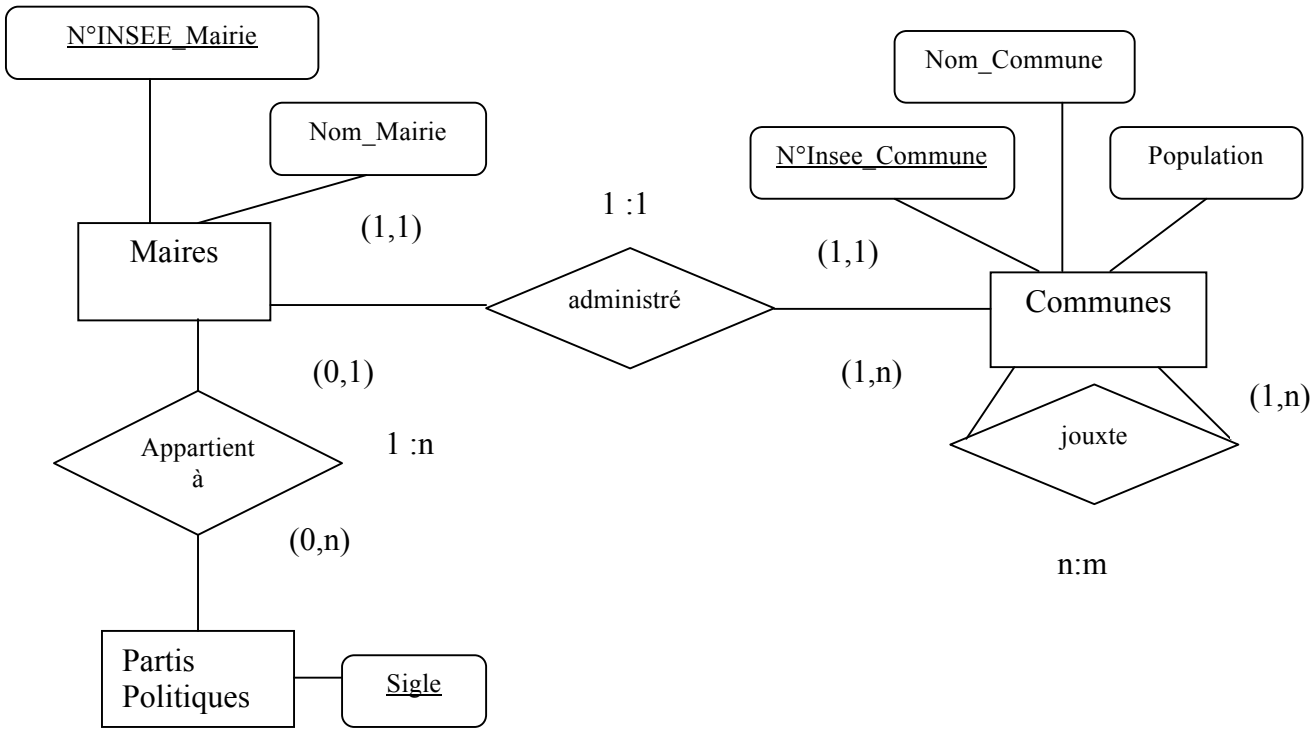
Transformez le schéma E/A de l'exercice 3 du Td numéro 1 & 2 en un modèle relationnel.

Exercice 3

Transformez le schéma E/A de l'exercice 4 du Td numéro 1 & 2 en un modèle relationnel.

Exercice 4

Transformez le schéma E/A suivant en un modèle relationnel



3.24. TD numéro 4
Dépendances fonctionnelles, Algèbre relationnel

Exercice 1

Soit une relation de schéma R(A,B,C,D,E) munie des dépendances fonctionnelles F={B→A ; B→C ;B→D ;B→ACDE}, que pouvez vous-affirmer concernant l'attribut B ?

Exercice 2

Pour une relation nommée Bourse, on a l'ensemble F des dépendances fonctionnelles F={NumContrat→Agence ;NumContrat→Client ;
NumContrat→NomProduit ; NomProduit→TypeProduit ;
NomProduit→DuréePlacementProduit ;
TypeProduit→DuréePlacementProduit}

a) Donner une représentation graphique de F (un graphe)
b) Calculer la fermeture transitive F^{+}_{t} de F et augmenter le graphe des relations obtenues

Exercice 3

Considérons l'ensemble F={A→B,C ; A→H ; B→H ; C,G→H ; C,G→I ; A,B→I }, démontrer que A,G→I en calculant la fermeture de {AG}. Vous utiliserez l'algorithme suivant qui permet d'obtenir tous les attributs déterminés par un ensemble d'attributs X relativement à l'ensemble des dépendances fonctionnelles F.

Résultat=X
Tant que résultat évolue
Pour chaque DF Y→Z ∈ F
Si Y ⊆ résultat alors résultat = résultat ∪ Z

Exercice 4

Déf : La couverture minimale d'un ensemble F de DF est l'ensemble de DF élémentaires vérifiant les propriétés suivantes :

- Toute DF élémentaire est dans F^{+}
- Aucune dépendance de F n'est redondante

Déf : Une DF X→A est élémentaire si A ne dépend pas d'un sous-ensemble d'attributs de X c'est à dire si ∄X′ ⊂ X/ X′→A

Soit l'ensemble de dépendances fonctionnelles F={A→B ; B,C→D ; D→E ; A,C→D ; A,C→E}

a) Ces dépendances sont-elles élémentaires ?
b) Y-a-t-il des DF redondantes ?
c) Quelle est la couverture minimale de F ?

Exercice 5

Soit le modèle relationnel suivant relatif à une base de données sur des représentations musicales :

REPRESENTATION(n°représentation, titre_représentation, lieu)
MUSICIEN(nom, *n°représentation*)
PROGRAMMER(date, *n°représentation*, tarif)

Vous formulerez en algèbre relationnel, les requêtes suivantes :

a) Donner la liste des titres des représentations
b) Donner la liste des titres représentations ayant lieu à l'opéra bastille
c) Donner la liste des noms des musiciens et des titres des représentations auxquelles ils participent
d) Donner la liste des titres des représentations, les lieux et les tarifs pour la journée du 14/09/1996

Exercice 6

Soit le modèle relationnel suivant relatif à la gestion des étapes de contre la montre du tour de France :

EQUIPE(CodeEquipe, NomEquipe, DirecteurSportif)
COUREUR(NuméroCoureur, NomCoureur, *CodeEquipe*, *CodePays*)
PAYS(CodePays, NomPays)
TYPE_ETAPE(CodeType, LibelléType)
ETAPE(NuméroEtape, DateEtape, VilleDép, VilleArr, NbKm, *CodeType*)
PARTICIPER(*NuméroCoureur*, *NuméroEtape*, TempsRéalisé)
ATTRIBUER_BONIFICATION(*NuméroEtape*, Km, Rang, NbSecondes, *NuméroCoureur*)

Vous formulerez en algèbre relationnel, les requêtes suivantes :

a) Quelle est la composition de l'équipe Festina (Numéro, nom et pays des coureurs) ?
b) Quels sont les noms des coureurs qui n'ont pas obtenu de bonifications ?

3.25. TD numéro 5
Formes Normales, Normalisation

Exercice 1

Soit la relation `PRETK7(NumCli, NomCli, AdrCli, NumK7, TiK7, DatePrêt)`

a) Quelles sont les DF entre les attributs de cette relation ?
b) Quelle est la clé de cette relation ?
c) Est-elle en 1FN ? en 2FN ?
d) Décomposer `PRETK7` en 3 relations en 2FN

Exercice 2

Soit la relation `FILM(N°Exploitation, Titre, Réalisateur)` et les dépendances fonctionnelles
`F={ N°Exploitation→Titre ; Titre→Réalisateur }`

a) Sous quelles formes normales se trouve cette relation ?
b) Proposez une décomposition en 3FN qui conserve les DF

Exercice 3

Algorithme de décomposition en 3FN :

1) Rechercher une couverture minimale `G` de D
2) Partitionner `G` en groupes de dépendances G_i ayant même partie gauche
3) Fusionner les groupes G_i possédant des parties gauches X_i et X_j équivalentes, c'est à dire ceux pour lesquels on a $X_i \rightarrow X_j$ et $X_j \rightarrow X_i$ avec $X_i \in G_i$ et $X_j \in G_j$
4) Associer à chaque groupe G_i un schéma $R_i=<U_i , D_i>$. Autrement dit, construire une relation R_i pour chaque G_i dont la clé est la partie gauche des DF de D_i et les constituants non clés est la partie droite des DF de D_i
5) Si aucun schéma ne contient une clé `K` de `R`, créer un schéma supplémentaire $<K,\emptyset>$

On considère la relation `R(A,B,C)` avec l'ensemble des DF `F={A→B ; B→C }`

a) Quelles est la clé primaire de `R` ?
b) Quelles est sa Forme Normale ?
c) Proposer une décomposition 3FN de `R` sans perte d'informations

Exercice 4

Soit la relation `R(A, B ,C, D ,E, F)` munie des dépendances fonctionnelles suivantes

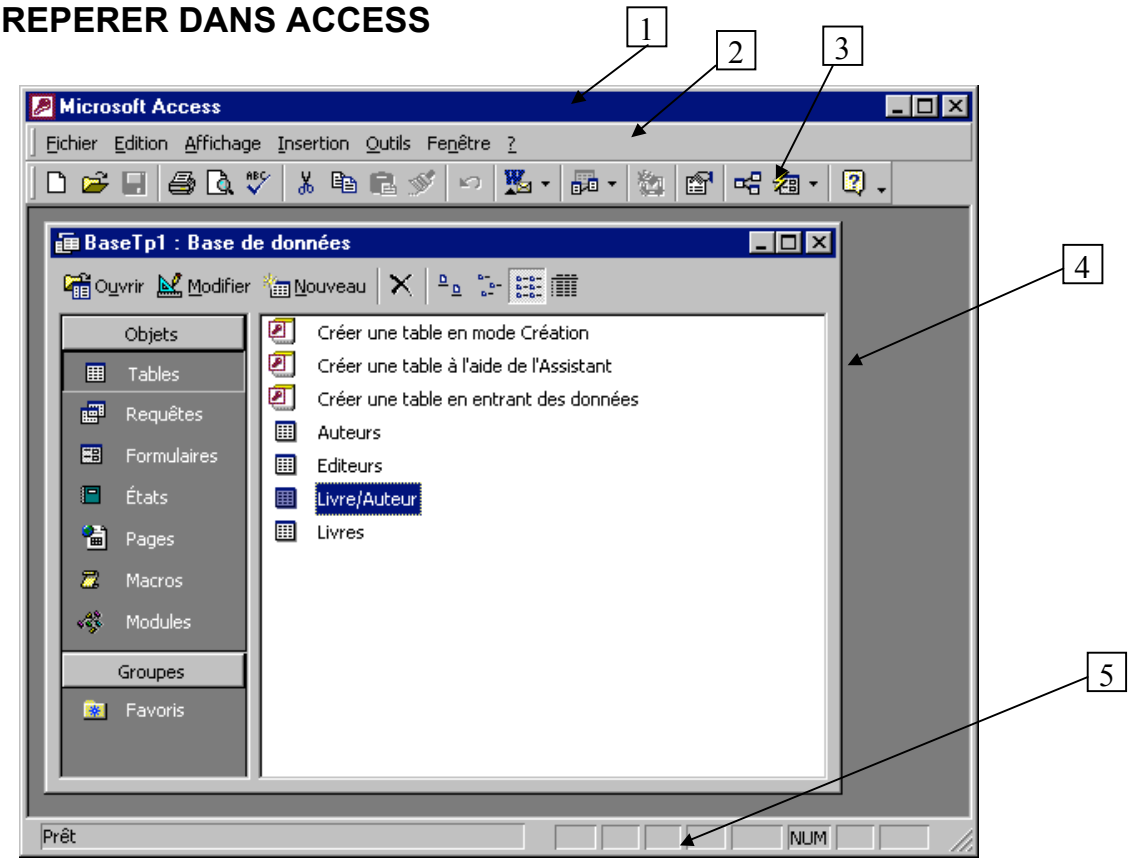
`F={A→B ; B→A ; A→F ; A,C→D ; C→E ;}`

Normalisez la relation en 3FN.

3.26. TP Numéro 1
INITITIATION A ACCESS – OBJETS MANUIPULES – CREATION DE TABLES ET DE FORMULAIRES

☞ **Pour lancer ACCESS** : ouvrir le menu « Démarrer → Programmes →Microsoft Office» puis sélectionner « Microsoft Access »

SE REPERER DANS ACCESS



1. Barre de titre
2. Barre de menu
3. Barre d'outils « Base de données »
4. Fenêtre d'une base de données
5. Barre d'état

SE REPERER DANS LA BARRE D'OUTILS « Base de données »

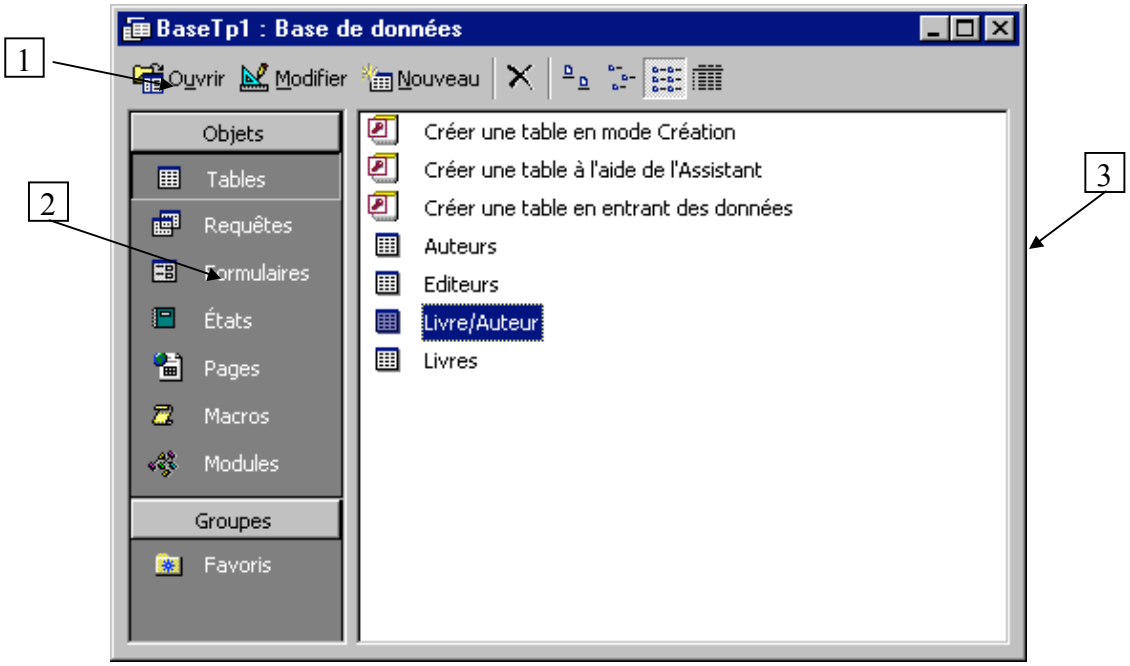
Cette barre permet d'activer les commandes les plus utilisées sans avoir à passer par le menu (elle présente des **raccourcis**). Pour connaître la signification d'un bouton, il suffit de placer le pointeur de la souris dessus et d'attendre. Une « **info-bulle** » apparaît alors, donnant une brève description. Une explication plus longue s'affiche aussi dans la **barre d'état**.

☞ Donnez la signification des 18 boutons de la barre d'outils « **Base de données** ».

1-	7-	13-
2-	8-	14-
3-	9-	15-
4-	10-	16-
5-	11-	17-
6-	12-	18-

Remarque : Contrairement à ***Word***, le bouton « *Annuler* » n'est pas une liste déroulante ; **ils ne peut agir que sur la dernière action effectuée**.

SE REPERER DANS LA VISUALISATION D'UNE BASE DE DONNEES



Cette fenêtre (qui n'est active que lors de l'édition d'une BDD) permet la visualisation du contenu d'une base de données. Elle est divisée en trois parties : une barre de menu (1), un menu de sélection (2) et une zone de visualisation (3).

OBJETS MANIPULES

ACCESS est un SGBD relationnel qui permet d'organiser, de gérer et d'exploiter des données enregistrées dans des tables grâce au maintien de « *relations* » entre les tables. Ces notions sont directement liées à celle de base de données relationnelle. Rappelons quelques définitions :

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

La table : C'est un objet qui stocke les données dans des lignes (enregistrements) et des colonnes (champs).

Le champ (ou colonne) : C'est l'unité d'information dans une table. Une table est constituée de différents champs. Chaque champ d'une table porte un nom différent ; Attention, certains SGBDR n'admettent pas d'espaces dans les noms de champs.

L'enregistrement (ou la ligne) : C'est l'ensemble des données relatives à la même information.

Clé primaire : C'est un champ d'identification qui permet d'identifier de façon unique chaque enregistrement. Cela permet de retrouver de façon non équivoque n'importe quel enregistrement dans une table. Il vaut mieux utiliser un seul champ et non une combinaison de champs pour réaliser une clé primaire car ce genre de clé complexe est difficile à maintenir, prend plus de place dans la base de données et fonctionne mal dans ACCESS.

Un SGBDR tel qu'ACCESS manipule et gère les données grâce à plusieurs objets : les tables, les formulaires, les requêtes et les états.

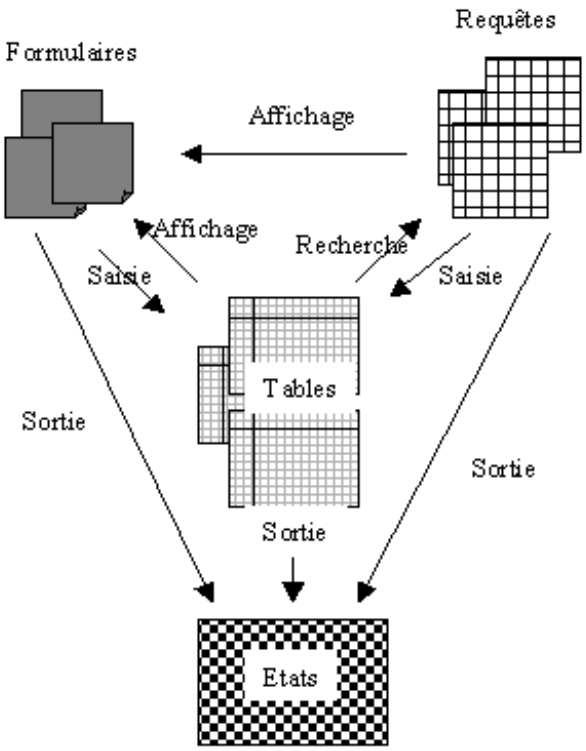
Les tables constituent toujours le cœur des bases de données ACCESS. Les formulaires, requêtes et les états peuvent se rapporter à des tables et ont pour rôle de traiter, de compléter, d'afficher ou d'imprimer les données enregistrées.

Un formulaire dépend des données d'une ou plusieurs tables ou requêtes. Normalement, on utilise les formulaires pour saisir des données dans une table. C'est un objet utilisé pour la saisie, la modification et l'affichage des enregistrements d'une table.

Une requête se base sur des tables ou d'autres requêtes. C'est un objet utilisé pour poser des questions à partir d'un jeu de critères sur des données stockées dans une ou plusieurs tables. Les données extraites des requêtes sont mises à jour lors de chaque nouvelle exécution de la requête.

Un état se rapporte à des tables ou à des requêtes. Il ne sert pas au traitement des données, mais à leur représentation. Il permet de calculer des valeurs de plusieurs façons différentes. Il est impossible de modifier les données d'un état. C'est un objet utilisé pour l'impression personnalisée d'enregistrements d'une ou plusieurs tables ou requêtes.

Les relations entre les objets sont représentées de manière simplifiée sur l'image suivante.



ETAPES DE MISE EN ŒUVRE SOUS ACCESS

La création et la gestion d'une base de données sous ACCESS se fait en plusieurs étapes :

1. Création d'une nouvelle base de données (l'extension est automatiquement .MDB)
2. Création des tables en indiquant le format de chaque champ
3. Création des relations entre les tables
4. Création des formulaires de saisie
5. Création de requêtes pour interroger la base de données
6. Création d'états pour faire le point sur le contenu de la BDD

CREATION D'UNE NOUVELLE BASE DE DONNEES

☞ Démarrez l'application MICROSOFT ACCESS

☞ Demandez la création d'une nouvelle base de données sans utiliser l'assistant

☞ Enregistrez la BDD sous le nom « NomPrénom_TP1.mdb » dans le répertoire *c:\Temp* si vous n'êtes pas en réseau sinon dans votre compte réseau.

Création des tables Auteurs, Editeurs et Livres

Définition des tables

Auteurs : Table

AuID	AuTitre	AuNom	AuDateNaiss	AuCodePostal	AuVille	AuTel	AuObservation
1	Monsieur	Austen	05/10/00	50000	Saint-Lô	02-33-45-27-61	ceci est le texte
3	Madame	Jones	02/01/72	14000	Caen	02-31-45-29-20	
4	Madame	Snoopy	09/08/73	50100	Cherbourg	02-33-20-70-00	
5	Monsieur	Grumpy	15/12/54	27000	Evreux	02-44-26-95-85	
6	Mademoisell	Sleepy	12/12/96	76000	Rouen	02-36-58-90-10	
7	Monsieur	Melville	22/05/82	35000	Rennes	02-96-85-74-12	
8	Monsieur	Homer	06/02/02	53000	Laval	02-36-45-47-89	
9	Madame	Roman	15/03/85	14000	Caen	02-31-45-87-95	
10	Mademoisell	Shakespeare	14/05/01	75000	Paris	01-20-59-87-10	
11	Madame	Joyce	17/09/74	45000	Orleans	03-56-98-74-52	
12	Monsieur	Spenser	06/05/92	61000	Alençon	02-36-98-74-15	
13	Monsieur	Mill	03/04/82	14000	Paris	01-25-87-53-69	
14	Madame	Smith	03/11/96	50000	Saint-Lô	02-33-45-96-87	
(NuméroAuto)				00000			

Enr : 8 sur 13

Editeurs : Table

PubID	PubNom	PubTel
1	BigHouse	02-25-69-87-02
2	Alpha Press	02-35-69-87-10
3	Small House	06-58-92-01-71
(NuméroAuto)		

Enr : 1 sur 3

Livres : Table

ISBN	Titre	Prix	PubId	Couleur
0-10-34567-9	Hiad	25,00 F	1	☑
0-11-34567-9	Jane Eyre	49,00 F	2	☐
0-12-33343-3	On Liberty	250,00 F	3	☑
0-12-34567-0	KingLear	34,00 F	1	☐
0-12-34567-9	Ulysses	49,00 F	1	☑
0-32-32312-1	Moby Dick	34,00 F	3	☐
0-55-12345-9	Ballon	22,95 F	2	☑
0-55-55555-9	Java Design	12,00 F	2	☐
0-91-04567-5	Hamlet	20,00 F	3	☑
0-91-33567-7	Faerie Queene	15,00 F	1	☑
0-99-77777-7	MacBeth	49,00 F	2	☐
0-99-99999-9	Emma	20,00 F	3	☐
1-11-11111-1	C++	29,95 F	1	☑
1-22-23370-0	Visual Basic	25,00 F	1	☑
		0,00 F	0	☐

Enr : 1 sur 14

Création des tables

☞ Pour créer une nouvelle table, dans l'onglet « *Table* », demandez à créer une nouvelle table (bouton *Nouveau*) en utilisant le mode (**Mode Création**)

☞ Pour chacune des tables, remplissez la colonne « Nom du Champ » avec les noms des champs des tables. Vous définirez comme clé primaire les champs *AuID*, *PubID* et *ISBN* en cliquant avec le bouton droit de la souris sur la ligne.

☞ Dans la colonne « *Type de données* », choisissez un format compatible avec les données à stocker. Pour les champs *AuID* et *PubID*, vous choisirez un type *NumeroAuto* pour créer une numérotation automatique. Pour le champ ISBN vous

prendrez un type texte. Dans l'onglet « *général* » situé en bas de la fenêtre, choisissez quand c'est possible un masque de saisie (pour les téléphones, le code postal, la date de naissance, le n° ISBN). Définissez également les propriétés des champs quand c'est possible.

☞ Identifiez rapidement les options de type de données ainsi que les propriétés des champs

Les listes de choix

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

Le champ AuTitre de la table Auteurs

☞ Dans la table *Auteurs*, on dispose d'un champ *AuTitre* qui donne le statut de l'auteur parmi Monsieur, Madame, Mademoiselle. Nous allons retourner dans la table *Auteurs* et modifier le type de données du champ « *AuTitre* » qui contenir le titre du client. Au lieu de saisir à chaque fois le mot en entier, nous allons créer une liste de choix qui proposera ces trois titres et il suffira de cliquer sur l'un d'entre eux pour remplir automatiquement le champ.

☞ Ouvrez la table *Auteurs* en modification et choisissez une liste de choix, sélectionnez parmi les valeurs choisies que vous rentrerez comme étant les trois valeurs du titre. Donnez ensuite l'étiquette « *AuTitre* » à votre liste de choix.

☞ Ouvrez la table Auteurs en édition et saisissez les données de la table en utilisant la liste de choix.

Le champ PubID de la table Livres

☞Saisissez les données de la table *Editeurs*

☞ Un livre contient le numéro de l'éditeur qui publie le livre. Lorsque l'on a saisi un livre et que l'on a le nom de l'éditeur, il faut se souvenir de son numéro pour le taper. Une autre façon plus performante de faire consiste à insérer dans la table « *Livres* » une liste de choix qui prend ses valeurs dans la table « *Editeurs* ». La liste de choix va proposer la liste de tous les éditeurs se trouvant dans la table « *Editeurs* », il suffira de cliquer sur le nom de l'un d'entre eux pour que son numéro soit automatiquement entré dans le champ *PubID* de la table « *Auteurs* »

☞ Ouvrez la table « *Livres* » et choisissez « Liste de choix » pour le champ *PubID*, demandez à récupérer les données de la table « *Editeurs* »

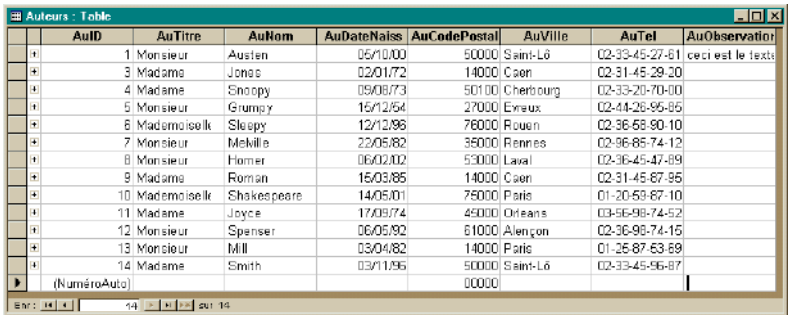
☞ Parmi les champs proposés, choisissez les champs *PubID* et *PubNom*.

☞ Vous choisissez la largeur des colonnes et à l'étape suivante, on vous demande ce que vous voulez faire du résultat provenant de la liste déroulante. Vous devez demander à stocker le résultat provenant du champ *PubID*, c'est à dire la clé primaire de la table « *Editeurs* » qui devient donc une clé étrangère de la table « *Livres* ».

☞ Saisissez les données de la table « *Livres* » en utilisant la liste de choix

Ajout de nouveaux enregistrements

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



☞ Visualisez le contenu de la table « *Auteurs* » à l'aide des icônes fléchés de navigation

☞ Cachez la colonne *AuObservations* grâce à l'option « *Afficher les colonnes* » du Menu « *Format* ». Cela fait juste disparaître le champ de l'affichage pour la saisie mais ne le supprime pas de la table. Rendez ensuite la colonne à nouveau visible.

☞ On peut également "figer" une colonne : Figez la colonne du numéro d'identifiant d'un auteur. Cela sert à avoir en permanence le numéro de l'auteur. Cela se fait par « *Figer les colonnes* » dans le Menu « *Format* ». Ainsi le numéro de l'auteur sera toujours visible même en déplaçant l'ascenseur.

Tri et Filtrage

Pour trier la table sur un champ, on se positionne sur le champ en question et on utilise un des deux icônes de la barre d'outils qui apparaît lorsque l'on édite une table en ajout d'enregistrements :  .

☞ Triez la table « *Auteurs* » par ordre alphabétique sur le champ *AuNom*.

Les filtres permettent de limiter de façon temporaire les enregistrements affichés dans la table. On peut filtrer sur un critère (un champ), il suffit de sélectionner la valeur du champ et de cliquer sur  . Pour faire réapparaître tous les enregistrements, on clique sur  .

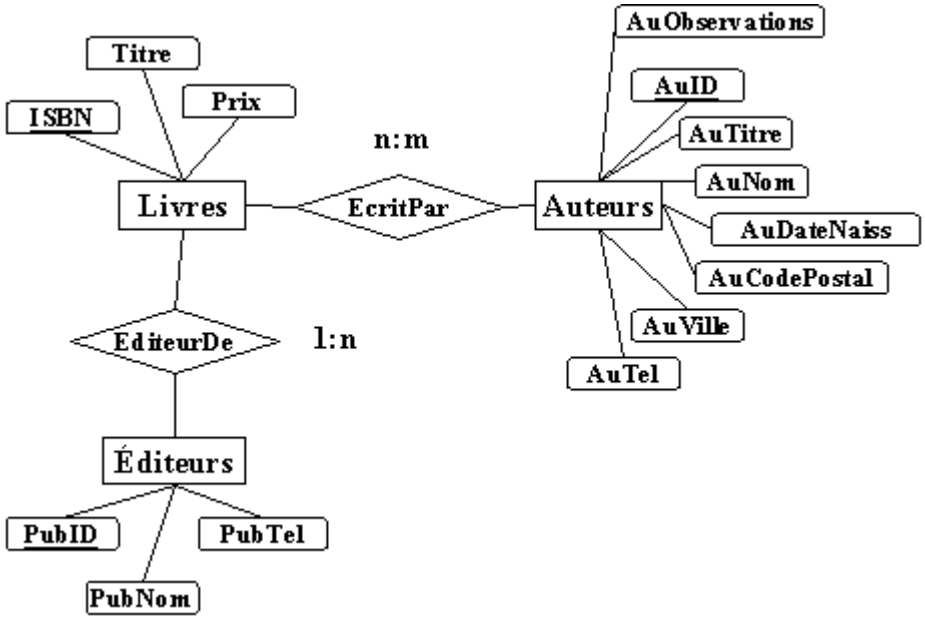
☞ Filtrez la table « *Auteurs* » sur la ville de Caen

Pour filtrer sur plusieurs critères, on clique sur l'icône  et un enregistrement vide s'affiche. Sélectionnez les livres de la table « *Livres* » qui valent *25,00F* et qui sont édités par l'éditeur *BigHouse*. En cliquant sur l'onglet « OU » qui se trouve en bas de la définition du filtre, on peut définir les critères d'une deuxième condition. Nous voulons maintenant les livres de la table « *Livres* » qui valent *25,00F* qui sont édités par l'éditeur *BigHouse* ou les livres en couleur édités par « *Alpha Press* ». Pour obtenir le résultat du filtre on clique sur l'icône  .

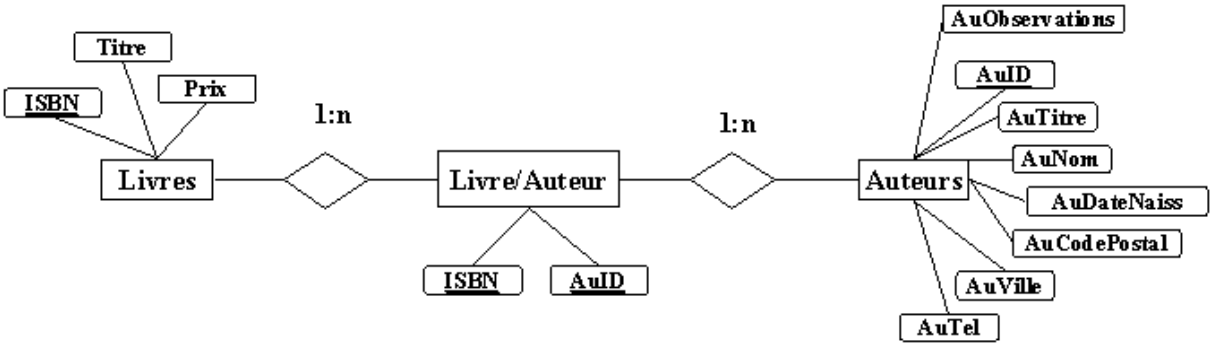
Pour rechercher des enregistrements, on utilise l'icône  : cherchez l'auteur Snoopy dans la table « *Auteurs* »

Création des relations entre les tables

Nous allons maintenant relier les tables entre elles en respectant le schéma obtenu après la phase d'analyse et conception qui nous a fourni le schéma E/A suivant. Dans ACCESS on ne peut implanter que des relations de type un à un (1:1) et un à plusieurs (1:n). Il sera vu en cours et TD comment passer du modèle E/A au modèle relationnel en transformant les relations et entités.

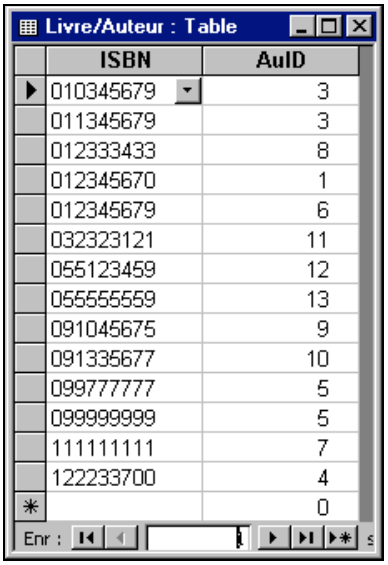


Les relations de type 1:n entraînent l'ajout de la clé primaire de l'entité de plus grande cardinalité à celle de plus petite (ici *PubID* doit donc faire partie de la table « *Livres* »). Les relations de type plusieurs à plusieurs ne peutventse faire comme pour les relation un à un car cela va encombrer la base de données d'informations redondantes. L'approche correcte pour établir une relation de type plusieurs à plusieurs consiste à ajouter un nouveau schéma de table, de manière à scinder la relation en deux relations de type un à plusieurs. Dans notre cas on ajoutera un schéma de table « *Livre/Auteur* » dont les attributs seront constituées des clés étrangères *ISBN* et *AuID*.



☞ Pour visualiser les liens entre les tables, il suffit de cliquer sur  ou de faire « *Outils → Relations*». Le lien entre les tables « *Livres* » et « *Editeurs* » existe déjà, il a été créé automatiquement lors de la création de la liste de choix associée à *PubID* de la table « *Livres* » qui met les deux tables en relation. Les clés primaires des tables apparaissent en gras.

☞ Créez la table « *Livre/Auteur* » avec les types numérique et texte pour chacun des attributs *AuID* et *ISBN* sans spécifier une quelconque liste de choix.



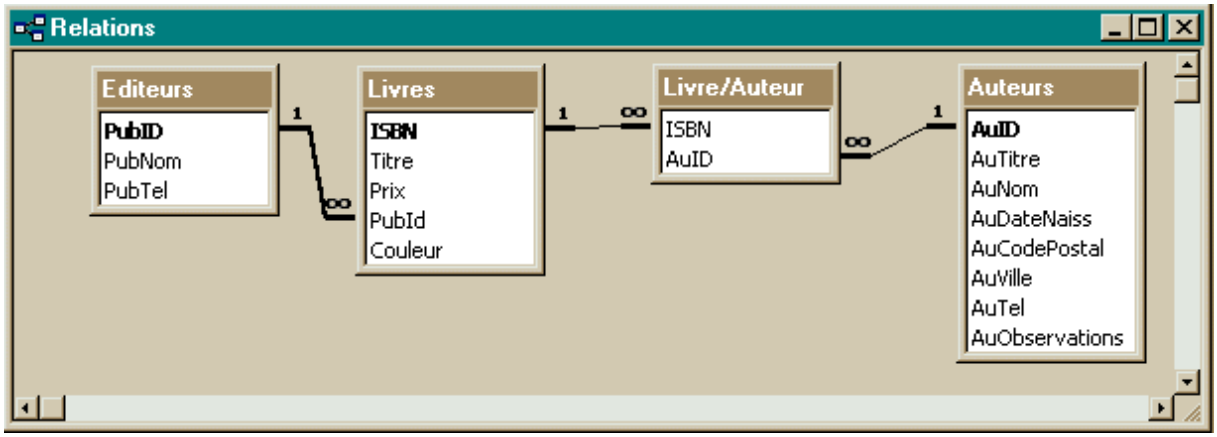
Livre/Auteur : Table

ISBN	AuID
010345679	3
011345679	3
012333433	8
012345670	1
012345679	6
032323121	11
055123459	12
055555559	13
091045675	9
091335677	10
099777777	5
099999999	5
111111111	7
122233700	4
	0

Enr : 1

☞ Liez les deux tables *Livres* et *Livre/Auteur* à la souris en effectuant une relation entre le champ *ISBN* de chacune des deux tables. Ne cochez pas pour l'instant l'option « appliquer l'intégrité référentielle ». Cela aura pour effet de créer le lien entre les tables. Fermez la fenêtre « *Relations* » en enregistrant les modifications.
☞ Notez dans la table « *Livre/Auteur* » un numéro *ISBN* inexistant dans la table « *Livres* » : ce numéro d'*ISBN* n'existe pas et la référence est boiteuse. Fermez la table et enregistrez la. Normalement ACCESS ne doit pas vous alerter.

☞ Nous allons corriger le lien existant entre les tables « *Livres* » et « *Livre/Auteur*» afin qu'il respecte l'intégrité référentielle. Modifiez le lien existant entre les tables « *Livres* » et « *Livre/Auteur* » (clic droit sur le lien puis « *Modifier une relation* ») et cochez l'option appliquer l'intégrité référentielle : vous devriez obtenir un message d'erreur indiquant que votre action est illégale. L'intégrité référentielle est l'obligation que chaque valeur de la clé étrangère soit une valeur existante de la clé primaire associée : il faut s'assurer que lorsque l'on associe deux tables par l'intermédiaire d'une clé primaire, cela garanti le bon fonctionnement de la BDD.
☞ Supprimez de la table « *Livre/Auteur* » la référence boiteuse puis modifiez la table « *Livre/Auteur* » pour avoir deux listes de choix pour chacun des attributs *AuID* et *ISBN*. Cela aura pour effet de créer les liens entre les tables, modifiez les pour qu'ils assurent l'intégrité référentielle.
☞ Finalement, Les relations que vous devrez obtenir dans ACCESS sont les suivantes :



Création de formulaires

Un formulaire va nous permettre d'afficher et de modifier le contenu d'une table de façon bien plus agréable que le mode « feuille de données » qui ne permet qu'un affichage en lignes et en colonnes. De plus ce dernier mode ne permet l'affichage et la modification d'informations ne provenant que d'une seule table, le formulaire va nous permettre de manipuler au même endroit des informations provenant de plusieurs tables.

☞ Nous allons créer le formulaire associé à la table « *Auteurs* ». Pour créer un formulaire, on clique sur l'onglet « Formulaire » puis on utilise l'assistant.

☞ Sélectionnez tous les attributs de la relation sauf *AuObservations*, prenez une présentation en colonne simple avec un style industriel. Appelez le formulaire « *Formulaire Auteurs* ».

☞ Passez en mode modification du formulaire, une boîte à outils contenant des contrôles à insérer dans le formulaire apparaît. Identifiez chacun des contrôles.

☞ Modifiez ensuite le formulaire pour rajouter une zone de texte d'intitulé *AuObservations* puis à l'aide d'un clic droit de la souris et « *Propriétés* », précisez l'attribut source du contrôle (*AuObservations*) et précisez que le contrôle possède une barre verticale.

☞ Insérez ensuite une image sur la droite du formulaire.

☞ Fermez le formulaire et double-cliquez dessus : le formulaire est affiché en mode saisie/affichage de la base de données. Pour se déplacer parmi les enregistrements du formulaire, on utilise les icônes fléchés en bas du formulaire.

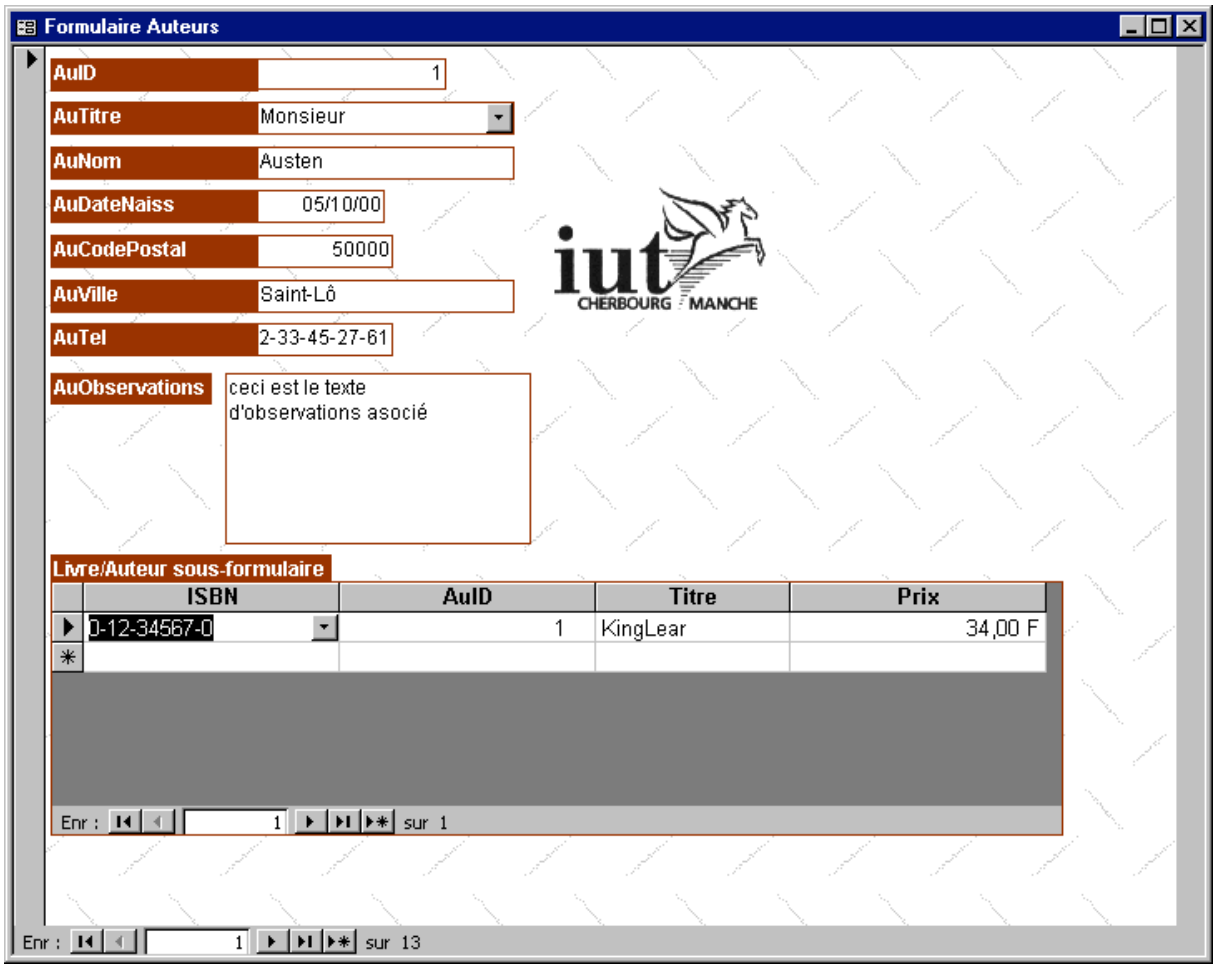
Les sous-formulaires

L'un des intérêts des formulaires est qu'ils permettent l'affichage et la modification d'informations ne provenant pas que d'une seule table. Nous allons rajouter au formulaire que nous venons de créer un sous-formulaire qui affichera des données

provenant d'une autre table. Grâce au contrôle « sous formulaire », nous allons donc pouvoir afficher des formulaires imbriqués dans d'autres.

☞ Dans notre cas, il serait intéressant d'avoir tous les livres écrits par chaque auteur sur chaque formulaire. Modifiez le formulaire « *Formulaire Auteurs* » et ajoutez sous la zone de texte un contrôle de sous-formulaire. Choisissez la table « *Livre/Auteur*» et sélectionnez tous les champs (à savoir *ISBN* et *AuID*), puis ajoutez les champs *Titre* et *Prix* de la table « *Livres* ». Il reste ensuite à préciser quels champs faisant le lien entre le formulaire et le sous-formulaire seront affichés : choisissez les champs vous-même et précisez que le lien se fait par le champ *AuID* (c'est l'identifiant de l'auteur qui sert à faire la relation avec les autres tables). Verrouillez enfin les données du sous-formulaire.

☞ Visualisez le résultat en utilisant le formulaire, pouvez vous modifier des valeurs du sous-formulaire ?

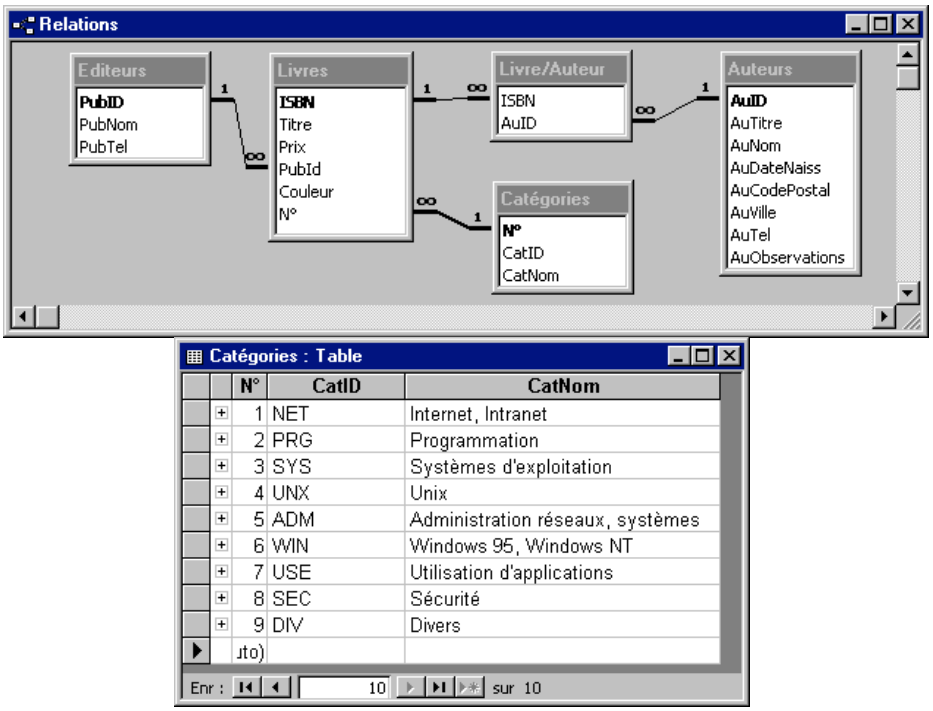


Conservez bien votre fichier pour pouvoir le réutiliser au prochain TP !

3.27. TP Numéro 2
CREATION D'INDEX — CREATION DE REQUETES — CREATION D'ETATS

Complément au TP numéro 1

Tout d'abord nous allons rajouter la table « *Catégories* » qui donnera pour chaque livre sa catégorie. Pour cela, vous devrez rajouter un champ nommé *CatID* dont la saisie se fera par une liste de choix. La relation qui existe entre les tables « *Livres* » et « *Catégories* » est de type un à plusieurs et assure l'intégrité référentielle.



Catégories : Table

	N°	CatID	CatNom
+	1	NET	Internet, Intranet
+	2	PRG	Programmation
+	3	SYS	Systèmes d'exploitation
+	4	UNX	Unix
+	5	ADM	Administration réseaux, systèmes
+	6	WIN	Windows 95, Windows NT
+	7	USE	Utilisation d'applications
+	8	SEC	Sécurité
+	9	DIV	Divers

Enr : 10 sur 10

Vous respecterez le contenu de la table « *Catégories* » et remplirez à votre gré le nouveau champ ajouté à la table « *Livres* ».

Les index

Comme les accès disque sont lents, il est nécessaire d'utiliser un fichier d'index qui va fournir un accès direct aux données dans un fichier de base de données. Pour optimiser les temps de recherche dans les tables, ACCESS (et d'ailleurs tous les SGBD) a besoin d'index. L'index permet à ACCESS de retrouver plus rapidement les enregistrements concernés ; il accélère aussi les opérations de tri.

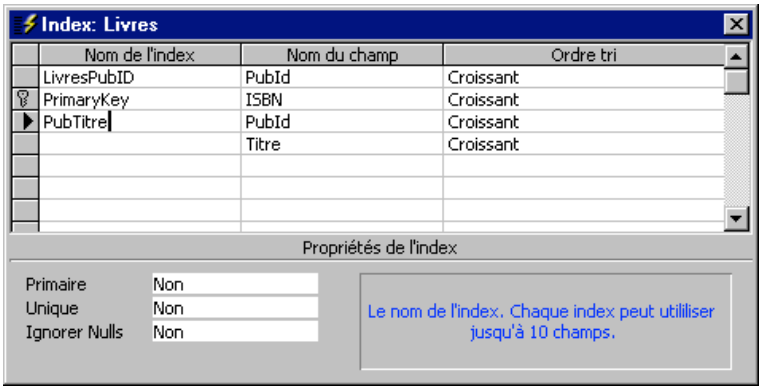
Les index fonctionnent avec une table comme la table des matières avec un livre. Lorsqu'on indexe sur un champ, ACCESS stocke hors de la table les valeurs de ce champ. Lorsque l'on effectue une recherche sur ce champ, plutôt que de le chercher dans la table, ACCESS va faire sa recherche dans l'index.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

Supposons que la table « Auteurs » soit indexée sur le nom de l'auteur, lorsque l'on cherchera un enregistrement de la table à partir de l'auteur, ACCESS parcourra

l'index et affichera l'enregistrement correspondant à l'index qu'il aura trouvé. Un index peut être constitué de plusieurs champs, cela accélérerait les recherches et les tris, mais ralentirait toutes les mises à jour car à chaque modification d'un champ indexé dans la table, il faut le modifier dans l'index.

☞ Pour voir les index d'une table donnée, ouvrez la table dans le mode création et choisissez *Index* depuis le menu *Affichage* ou bien cliquez sur le bouton  . Pour ajouter un index sur plus d'un attribut, vous devez entrer les divers attributs sur les lignes successives de la boite de dialogue *Index*.

☞ Ajoutez un index *PubTitre* basé sur les attributs *PubID* et *Titre* pour la table « *Livres* ». Cet index effectue l'indexation des entités LIVRES d'abord par *PubID* et ensuite par *Titre*.



Les requêtes

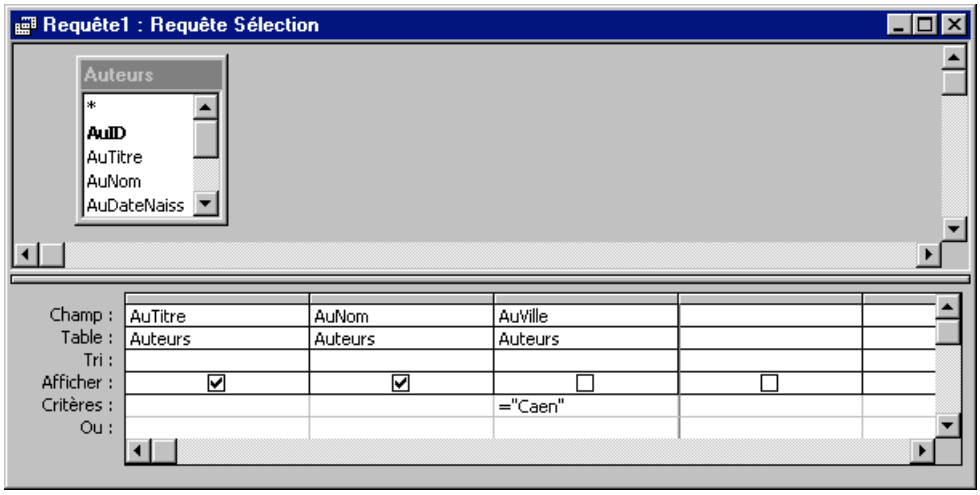
Les requêtes vont servir à afficher uniquement certaines données contenues dans les tables selon certains critères. Elles peuvent aussi faire des calculs sur vos données, ainsi que créer des sources de données pour les formulaires, les états ou même d'autres requêtes. Elles servent encore à modifier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

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

- La requête sélection
- La requête d'analyse croisée
- La requête action

Les requêtes sélection

☞ Pour créer une requête, cliquez sur l'onglet « *Requêtes* » puis sur le bouton « *Créer une requête en mode création* ». ACCESS nous affiche la liste des tables de la base. Nous choisissons ici les tables sur lesquelles vont porter la requête. Prenons la requête « *Liste des auteurs habitant la ville de Caen* ». Le requête va donc porter sur la table « *Auteurs* », on l'ajoute et on ferme. Faites en sorte d'avoir une requête qui affiche le titre, et le nom de tous les auteurs habitant à Caen.



☞ Nommez cette requête « *RequeteVille* » et Double-cliquez sur l'icône de votre requête pour voir le résultat.

☞ Recommencez les opérations précédentes pour créer une requête (nommée « *RequêteLivres* ») qui affiche l'*ISBN*, le *Titre* et le *Prix* d'un livre. Pour exécuter la requête en mode création, on clique sur l'icône  .

☞ Donnez la signification de chacune des lignes de définition de la requête

Les critères de sélection

Dans les requêtes de sélection , on peut utiliser les opérateurs =, <>, <, >, <=, >=, qui correspondent respectivement à égal, différent, inférieur, supérieur, inférieur ou égal, supérieur ou égal. Mais on peut également utiliser *Entre*, *Dans*, *Pas*.

☞ Créez les requêtes suivantes :

- Affichez par ordre décroissant de *Prix*, le *Titre* et *PubID* des livres dont le prix est entre 0 et 25 FR et dont l'éditeur n'est pas le numéro 1 (nommée « *RequêteSelection1* »)
- Affichez les numéros de téléphone et le nom de toutes les femmes mariés habitant à Saint-Lô et nées entre le 01 Janvier 80 et le 01 Janvier 99 (nommée « *RequêteSelection2* »)
- Affichez les livres dont le numéro ISBN commence par 1 et qui sont en couleur (nommée « *RequêteSelection3* »)

On peut utiliser des fonctions dans les critères de sélection. ACCESS met à notre disposition un très grand nombre de fonctions qui sont également utilisées dans les contrôles des formulaires.

☞ A l'aide de la fonction *Mois()*, créez une requête (nommée « *RequêteSelection4* ») qui affiche le nom des auteurs nés au mois de décembre. Vous pouvez utiliser le générateur d'expression en cliquant sur le bouton  .

Les critères de sélection que nous avons défini peuvent être multi-champs, ces champs étant séparés entre eux par des *ET*. On peut utiliser un critère de type *OU* pour les requêtes de la forme « *Liste des auteurs habitant à Paris ou Cherbourg* ». Les deux critères sont l'un sur la ligne « *critères* » et l'autre sur la ligne « *OU* ».

☞ Créez une requête qui affiche les noms des auteurs habitant Paris ou Cherbourg (nommée « *RequêteSelection5* »), et une autre requête affichant les titres des livres dont le prix est entre 0 et 25 FR ou entre 50 et 500 FR (nommée « *RequêteSelection6* »).

Lorsque vous exécutez une requête, vous pouvez faire en sorte qu'une « *boite de dialogue* » s'affiche pour demander le critère a appliquer. Cette démarche consiste à modifier le contenu de la ligne « *Critères* » de la requête et à placer la question entre crochets. Cela va nous servir à répondre aux requêtes du type « *Liste des clients qui habitent dans une ville donnée* », mais qu'on veuille entrer nous même la ville.

☞ Modifiez la requête « *RequeteVille* » en tapant dans la ligne « *Critères* » le texte suivant *=[Entrez la ville]*. Enregistrez la requête, exécutez la, entrez le nom d'une ville et constatez le résultat de votre action.

☞ Créez une requête (nommée « *RequêteSelection7* ») qui va afficher le titre, le numéro ISBN et l'éditeur des livres (le champ *PubID*) pour un prix donné, ces livres devant ne pas être en couleur.

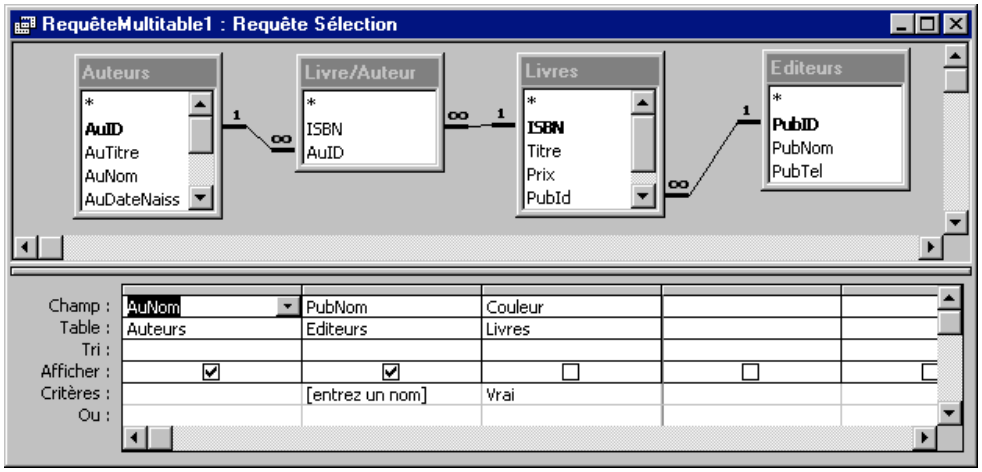
☞ Créez la requête (nommée « *RequêteSelection8* ») pour faire « *Affichez les livres dont le numéro ISBN commence par un nombre donné et qui sont en couleur* » : le numéro par lequel commence l'ISBN sera précisé par l'utilisateur.

Pour l'instant, nos requêtes ne portaient que sur une seule table, il est possible de faire des requêtes qui portent sur plusieurs tables simultanément. Dans ce cas, les requêtes peuvent être très complexes. Par exemple, nous voulons la liste des auteurs ayant un éditeur donné pour leurs livres et dont ces livres sont en couleur.

☞ Créez une nouvelle requête (nommée « *RequêteMultitable1* »)et choisissez toutes les tables entrant en jeu pour faire cette requête. Le nom de l'éditeur devra être saisi par l'utilisateur. Le bouton  est utile pour afficher la liste des tables de la base. Attention : lorsque l'on fait une requête portant sur plusieurs tables, il faut respecter deux règles :

- Toutes les tables doivent être reliées entre elles.
- Il ne doit pas y avoir de tables inutiles

Vous devriez obtenir une requête ressemblant à ce qui suit et vous pouvez remarquer que, une fois les bonnes tables installées, la requête est fort simple.



Les requêtes de regroupement

Jusqu'à présent nos requêtes nous permettaient de répondre à des questions du type : « *Liste des auteurs habitant Paris* », « *Liste des livres écrits par l'auteur n°1* ». Grâce aux fonctions de regroupement, nous allons pouvoir répondre à des questions du type : « *Combien d'auteurs habitent Paris* » ou « *Montant total des prix des livres par auteur* ».

☞ Pour créer ces deux dernières requêtes, nous allons utiliser le bouton  . En cliquant dessus, une nouvelle ligne « *Opérations* » apparaît dans la requête, c'est grâce à elle que nous allons faire nos opérations. Pour obtenir le nombre d'auteurs qui habitent à Paris, vous utiliserez les deux champs *AuID* et *AuVille*. Sur la ligne « *Opérations* », vous prendrez l'opération « *compte* » et l'opération « *où* » là où il y a un critère de sélection (*AuVille* égal à *Paris*). Le résultat de la requête doit être un unique champ non modifiable, la requête est nommée « *RequêteRegroupement1* ».

☞ Pour la deuxième requête, on veut par auteur, le montant total des prix de ses livres. A partir du moment ou l'on veut un résultat par catégorie, il y a regroupement et c'est l'opération choisie. La requête devra afficher le nom de l'auteur et ce champ devra donc être sélectionné. Trouvez quelle opération utiliser pour avoir une réponse correcte à la requête, la requête est nommée « *RequêteRegroupement2* ».

☞ Créez une requête (nommée « *RequêteRegroupement3* ») qui donne le nombre de livres édités pour chaque éditeur et une requête (nommée « *RequêteRegroupement4* ») qui donne le nombre d'auteurs différents pour chaque éditeur.

Les requêtes d'analyse croisée

Les requêtes d'analyse croisée permettent de répondre à des questions du type « *Qui a écrit combien de quoi ?* ». Elles retournent le résultat sous forme d'un tableau comportant des champs en abscisse et en ordonnée.

Exemple : Qui a écrit combien de livres de telle catégorie ?

	Qui	**Qui**
Quoi	Combien	Combien
Quoi	Combien	Combien

L'en-tête « *Qui* » va contenir le nom des auteurs, l'en-tête « *Quoi* » va contenir la liste des catégories et « *Combien* » va donner, pour chaque auteur, le nombre de livres écrit dans la catégorie.

☞ Créez une requête standard qui donne le nom des auteurs et le code d'une catégorie (*AuID* et *CatID* des deux tables « *Auteurs* » et « *Catégories* »). Transformez la requête en requête d'analyse croisée en allant dans le menu requête. Dans la ligne analyse, indiquez quel sera l'en-tête des colonnes, des lignes et rajoutez un champ qui donnera le nombre des livres de chaque catégorie. La requête sera nommée « *RequêteAnalyseCroisée1* ».

☞ Créez la requête d'analyse croisée permettant de répondre à « *combien chaque auteur a-t-il écrit de livres chez chaque éditeur ?*». La requête sera nommée « *RequêteAnalyseCroisée2* ».

Les requêtes d'Action

Les requêtes faites jusqu'à présent se contentent de retourner le résultat d'une sélection sous forme de table. Les requêtes *Action* vont permettre de créer des tables à partir du résultat d'un requête, ajouter des enregistrement à une table à partir du résultat d'une requête, mettre à jour une table, supprimer des enregistrements. Lors de la création d'une requête, vous pouvez utiliser le bouton  qui se trouve à gauche sur la barre d'outils « *Base de données* ». Il permet de vérifier le résultat d'une requête.

Les requêtes création

Une requête *Création* crée une table à partir des résultats qu'elle produit.

☞ Créez la requête « *RequêteLivres2* » basée sur la requête « *RequêteLivres* » comme une requête *Création* par le menu *Requête → Requête Création de Table*. Appelez la nouvelle table « *Livres2* » et essayez la requête ainsi modifiée.

Les requêtes Ajout

Une requête *Ajout* copie tout ou une partie des enregistrements d'une table à la fin d'une autre.

☞ Créez la requête « *RequêteLivres3* » (identique à « *RequêteLivres* ») en la transformant en une requête Ajout (menu *Requête* → *Requête Ajout*) et sélectionnez la table « *Livres2* » comme table cible. Rajoutez le critère que seuls les livres couleur devront être ajoutés à la base. Exécutez votre requête et vérifiez que cela a bien fonctionné.

Les requêtes de Mise à jour

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

☞ On veut augmenter de 10% le prix actuel des livres dont le prix est inférieur à 30Fr. Créez une requête de mise à jour (requête « *RequêteMiseAjour* »)qui va augmenter de 10% le prix des livres présents dans la table « *Livres2* ». Dans la case *Mise à jour* on précise l'augmentation de 10% soit : *[Prix]*1.1* ceci pour les produits dont le prix est inférieur à 30Fr (critère). Exécutez votre requête et vérifiez que cela a bien fonctionné.

Les requêtes Suppression

Les requêtes *Suppression* permettent de supprimer un groupe d'enregistrements qui répondent à un critère donné. ☞ On veut supprimer tous les livres de la table « *Livres2* » dont le prix est compris entre 0 et 30 FR, la requête sera nommée « *RequêteSuppression* » et sera crée grâce au menu *Requête* → *RequêteSuppression*.

Les états

Les états vont permettre l'impression d'enregistrements selon une présentation qui aura été définie au préalable. La création d'un état ressemble à celle d'un formulaire. Comparé à un formulaire, un état met à votre disposition plus de possibilités d'agencement et de présentation des données selon vos besoins.

Création d'un état simple

Pour créer rapidement des états, on utilise les états instantanés colonnes et tableau obtenus à l'aide du bouton *Nouveau*.

☞ Créez deux états différents (colonne et tableau) pour la table « *Auteurs* », analysez les différences entre chacun d'entre eux (place du titre de la table, des noms de champs, des enregistrements).

Les deux précédentes méthodes étant trop restrictives, nous allons créer un état à partir de l'assistant. Comme précédemment, on choisit à partir de quelle table va être créé l'état, puis on sélectionne les champs que l'on veut voir apparaître dans l'état.

☞ Créez un état à l'aide de l'assistant pour afficher tous les champs de la table « *Auteurs* » (sauf *AuObservations*) en utilisant un premier regroupement selon la ville puis sur le titre. Un regroupement signifie que, dans l'état qui va être généré, ACCESS va afficher d'abord le nom de la ville avec tous les auteurs habitant cette ville puis la ville suivante etc. Les noms seront triés par ordre croissant. Vous choisirez le modèle et le style par défaut de l'état. Vérifiez finalement que l'état obtenu correspond bien aux spécifications demandées.

☞ On peut également créer un état à partir d'une requête. Créez un état avec les mêmes spécifications que précédemment mais avec le résultat d'une requête qui permet d'afficher uniquement les champs *AuTitre*, *AuNom*, *AuTel* et *AuVille*.

Création d'un état avec plusieurs tables

Jusqu'à présent, nos états n'affichaient que le contenu d'un table, nous allons créer un état qui affiche la liste des auteurs avec, pour chaque auteur, la liste des livres qu'il a écrit.

☞ Utilisez l'assistant pour créer un état en sélectionnant les champs appropriés provenant des tables « *Auteurs* » et « *Livres* ». ACCESS s'aperçoit qu'à un auteur peut correspondre plusieurs livres et propose donc de générer un état qui va soit afficher tous les auteurs avec pour chaque auteur les livres écrits ou bien tous les livres avec pour chaque livre l'auteur qui l'a écrit. Choisissez de regrouper par auteur puis par prix, les titres des livres étant triés par ordre alphabétique.

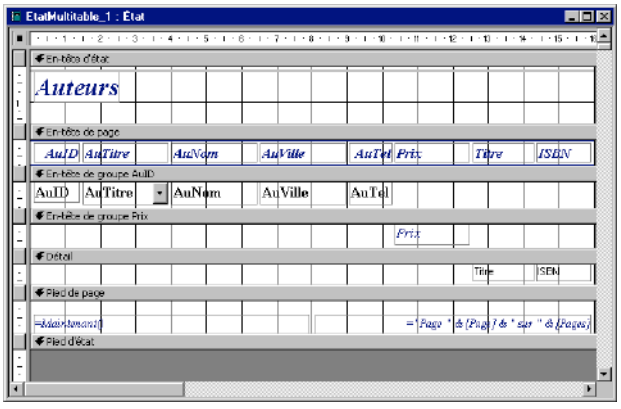
Modifier la structure d'un état

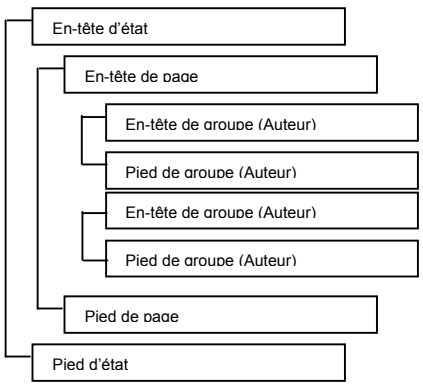
A présent, nous allons modifier l'état précédent pour avoir, pour chaque auteur, une ligne qui va indiquer combien il a écrit de livres. Ouvrez l'état en modification, vous pouvez constater qu'il est composé de plusieurs parties :

- En-tête de l'état : imprimé une seule fois au début de l'état
- En-tête de page : imprimé au début de chaque page
- En-tête de groupe : imprimé au début de chaque groupe (les auteurs)
- Le détail : chaque ligne de l'état (les livres)
- Le pied de page : imprimé à chaque fin de page
- Le pied d'état : imprimé à chaque fin d'état

On va afficher le nombre de livres écrits dans le pied de groupe auteur : à la fin d'un auteur, on affiche le nombre de livres qu'il a écrit. Cliquez sur  pour afficher le pied de groupe *Auteurs* Sélectionnez *Oui* pour afficher un pied de groupe. Dans le pied de groupe *Auteurs*, ajoutez un contrôle « *Zone de texte* » qui va nous permettre d'afficher le résultat d'un calcul (le calcul du nombre de livres écrits). Ajoutez *=compte(ISBN)* comme source du contrôle, le texte doit se changer *=compte([ISBN])*, ce qui atteste que le nom du champ *ISBN* a été reconnu. Vérifiez finalement que l'état affiche bien la valeur escomptée.

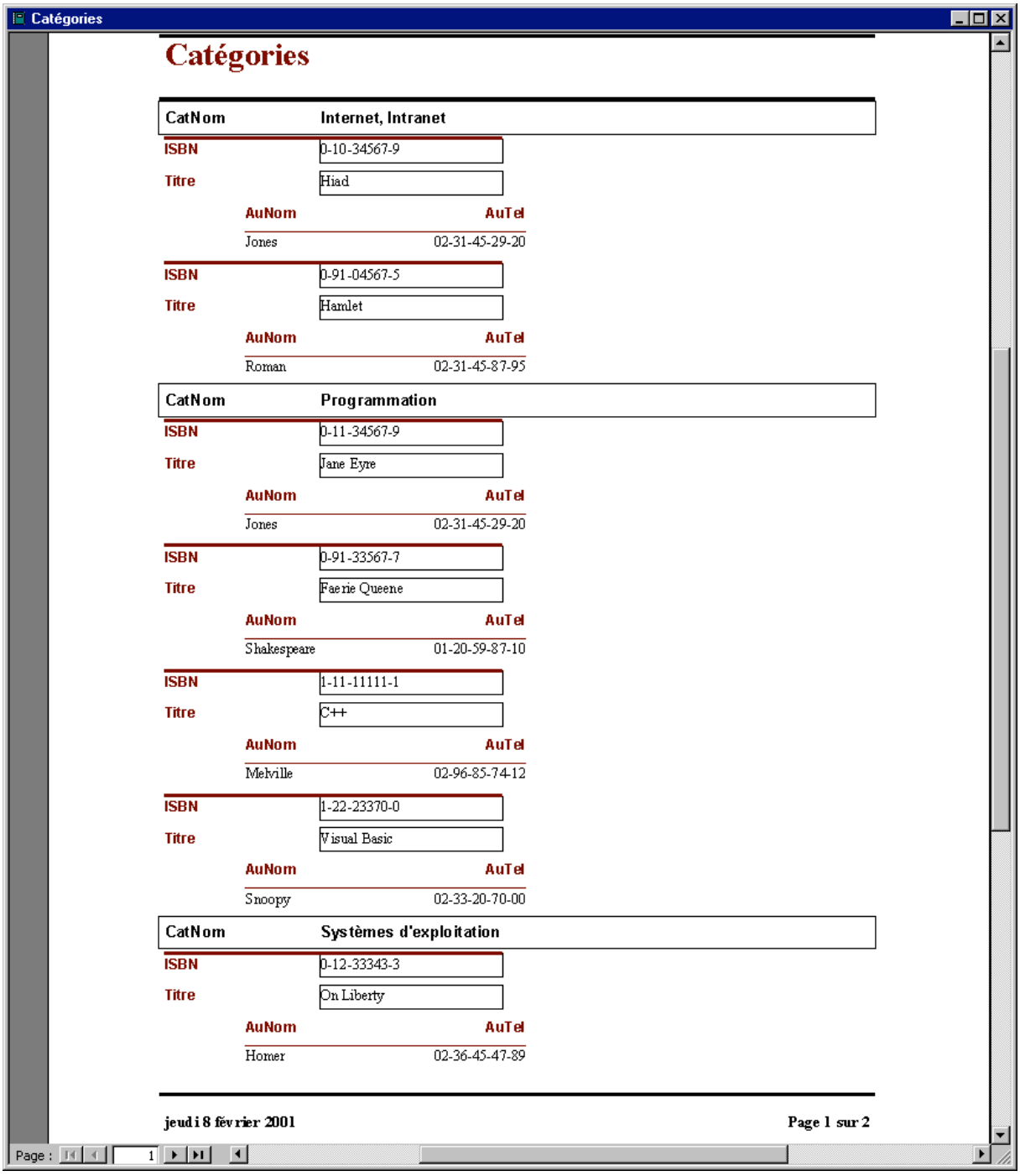
Remarque : Insérer des formules dans le zones « *En-tête ou pied d'état* » permet d'effectuer des calculs sur l'ensemble des éléments présentés.





☞ Déterminez les opérations à effectuer pour que l'on voit s'afficher pour chaque livre, le prix TTC (5,5% de majoration). Faites également s'afficher le nombre total de livres affiché pour l'état.

☞ Créez une requête qui va présenter un état semblable à celui de la page suivante.



Conservez bien votre fichier pour pouvoir le réutiliser au prochain TP !

3.28. TP Numéro 3
CREATION DE MACROS — ALGEBRE RELATIONNEL — EXERCICE RECAPITULATIF

Les Macros

Une macro permet d'automatiser certaines tâches d'**ACCESS**, elle peut exécuter une série de commandes qui auraient du être faites par l'utilisateur. Une macro est composée d'actions, chaque action correspond à une tâche. Certaines de ces actions plus complexes vous permettent d'afficher des boites de dialogue, de tester la réponse fournie par l'utilisateur. La méthode de création d'une macro est liée à deux facteurs importants :

- L'environnement de départ de la macro : certaines macros sont liées à un objet de la base de donnée, d'autres peuvent être exécutées quelle que soit la fenêtre active,
- L'événement qui va déclencher l'exécution de la macro : un clic sur un bouton, l'ouverture d'un formulaire, la valeur d'un contrôle etc.

Création d'une macro autonome

Une macro autonome n'est pas liée à un événement spécifique et peut être exécutée quelle que soit la fenêtre active. Nous allons créer une macro qui va ouvrir la table « *Auteurs* » et son formulaire.

☞ Dans la fenêtre principale d'**ACCESS**, cliquez sur l'onglet *Macro* puis sur *Nouveau*. La partie supérieure de la fenêtre (le tableau) est destinée aux différentes actions qui vont composer la macro. La plupart des actions ont des paramètres (par exemple l'action *OuvrirTable* qui ouvre une table et demande en paramètre le nom de la table à ouvrir). La partie inférieure de la fenêtre sert à indiquer les paramètres.

☞ Il existe un très grand nombre d'actions possibles, Utilisez l'aide pour avoir la description de chacune des actions possibles. Choisissez dans le menu déroulant l'action que l'on veut faire exécuter : ici nous voulons ouvrir une table, nous choisissons l'action « *OuvrirTable* ».

☞ Comme vous pouvez le constater, dans le champ « *NomTable* » apparaissent les noms de toutes les tables de la base, on choisit « *Auteurs* ». *Affichage* et *Mode données* servent respectivement à déterminer le mode d'affichage de la table (création ou édition) ainsi que le mode de saisie des données de la table (ajout, modification, lecture seule). Choisissez feuille de données et *Lecture Seule*.

Le tableau de la fenêtre « *Macros* » contient plusieurs lignes, on peut exécuter plusieurs actions les unes à la suite des autres dans la même macro.

☞ Maintenant, après avoir ouvert notre table « *Auteurs* », nous allons ouvrir le formulaire associé à cette table. Choisissez l'action « *OuvrirFormulaire* » avec le formulaire associé à la table (réalisé au TP n°1).

☞ Enregistrez votre macro sous le nom « *MacroAutonome1* », comme pour les requêtes, cliquez sur l'icône point d'exclamation, **ACCESS** exécutera toutes les lignes de la macro.

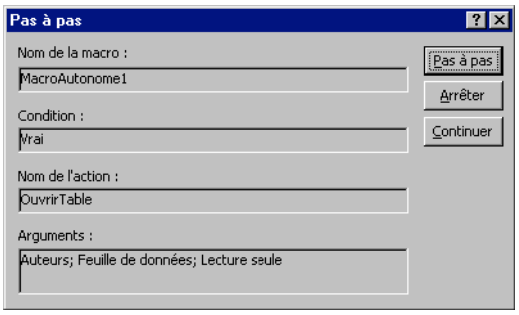
☞ Nous allons rajouter une troisième action à la macro : afficher une boîte de dialogue pour signaler la fin de l'exécution de la macro. Choisissez l'action

« *BoîteMsg* » avec Bip, information et « Fin de la macro » comme titre et message. Exécutez à nouveau la macro afin de vérifier que la boîte de dialogue s'affiche correctement.

Exécuter la macro pas à pas

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

1. Ouvrir la macro en mode création
2. Cliquer sur le bouton 
3. Exécuter ensuite la macro



ACCESS affiche alors une fenêtre décrivant la première action ainsi que ses paramètres. Vous pouvez alors continuer l'exécution de la macro en mode normal, arrêter l'exécution de la macro ou bien exécuter l'action en cours et passer à la suivante.

☞ Exécutez la macro « *MacroAutonome1* » en mode pas à pas.

Modifier une macro

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

☞ Modifiez la macro en ajoutant une action en tout premier, cette action devra transformer le curseur de la souris en sablier le temps de l'exécution de la macro. Rajoutez également une action qui permettra de réduire la fenêtre de la table « *Auteurs* ».

Création d'une macro associée à un formulaire

Certaines macros doivent être exécutées en réponse à un événement lié au formulaire (ouverture, fermeture), d'autres dépendant d'un événement lié à un contrôle du formulaire (clic, valeur du contrôle). Dans le premier cas, la macro doit être insérée dans la feuille de propriétés du formulaire et dans le second cas, elle doit apparaître dans la feuille de propriétés du contrôle. Supposons que nous voulions refuser l'enregistrement d'un auteur si la date de naissance de cet auteur n'a pas été saisie (nous pourrions indiquer dans les propriétés du champ *Null Interdit* : *Oui*). Quand va-t-on vérifier que le champ *AuDateNaiss* est vide ? Cela ne peut se faire que lorsque l'on passe à un nouvel enregistrement. L'événement « passer à un nouvel enregistrement » est lié au formulaire et pas au contrôle *Date*, l'événement sera donc associé au formulaire.

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

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

La liste des principaux événements qui peuvent être associés aux contrôles d'un formulaire :

Evénement	La macro doit s'exécuter
Sur Clc	Quand on clique sur le contrôle
Sur Double Clic	Quand on double clique sur le contrôle
Sur Entrée	Avant d'accéder à un contrôle
Sur Sortie	Lorsque l'on quitte un contrôle
Avant MAJ	Après avoir quitté un contrôle modifié et avant sa mise à jour

Après MAJ	Après avoir quitté un contrôle modifié et après sa mise à jour

Exécuter des actions en fonction de conditions

☞ On va créer une nouvelle macro pour vérifier la date. Une fois la fenêtre de création ouverte, cliquez sur le bouton  . Une nouvelle colonne apparaît dans la fenêtre de la macro, c'est la colonne « *Condition* ». C'est dans cette colonne qu'on va taper la condition qui décidera si oui ou non l'action de la macro doit s'exécuter. On veut qu'un message d'erreur s'affiche lorsque la date de la commande est vide, la condition pour que le message s'affiche est *[AuDateNaiss] Est Null*. L'action qui sera déclenchée si la condition s'avère vraie est l'affichage d'un message d'erreur. Vous utiliserez l'action « *BoîteMsg* » pour afficher une boîte de dialogue signalant une erreur de saisie.

Déplacer le curseur

Une fois le message d'erreur affiché, il va falloir positionner le curseur automatiquement sur le contrôle *Date*, pour que l'utilisateur puisse re-saisir la date. Il existe plusieurs actions pour déplacer le curseur (*AtteindreContrôle, AtteindrePage, AtteindreEnregistrement*).

☞ Nous devons déplacer le curseur dans le formulaire sur le contrôle du champ *Date*, nous utiliserons donc l'action « *AtteindreContrôle* ». Vous devrez préciser le nom du contrôle à atteindre pour cette action, pensez à le vérifier en affichant les propriétés du contrôle du formulaire.

☞ Vous devrez répéter la condition pour chacune des actions de la macro afin de préciser que toutes les actions ne se produisent que dans le cas où l'erreur serait apparue.

☞ Rajoutez maintenant l'action « *AnnulerEvénement* » entre « *BoîteMsg* » et « *AtteindreContrôle* ». Normalement, après avoir modifié le formulaire, **ACCESS** va mettre à jour les champs correspondants dans les tables liées à ce formulaire. Juste avant de faire cette mise à jour, **ACCESS** va appeler la macro associée à l'événement « *AvantMAJ* » ; La macro va être exécutée, puis la mise à jour sera faite. Il faut qu'on annule cette mise à jour sinon le message indiquant que la date est vide va être affiché puis **ACCESS** va sauvegarder les modifications et sauver dans la table « *Auteurs* » la date vide. L'action « *AnnulerEvénement* » annule la mise à jour qui allait être faite et la date ne sera pas sauvée.

☞ Enregistrez la macro sous le nom « *ErreurSaisieDate* ».

Affectation de la macro à l'évènement

☞ Maintenant que la macro est crée, on va l'affecter à l'événement « *AvantMAJ* » du formulaire « *Formulaire Auteurs* ». Pour afficher les propriétés d'un formulaire, cliquez avec le bouton droit sur le carré noir situé en haut à gauche du formulaire, puis dans le menu, cliquez sur *Propriétés*. Choisissez l'événement auquel vous affectez la macro « *ErreurSaisieDate* ». Une fois la macro associée, il suffit d'entrer un auteur pour lequel on omet de saisir la date de naissance pour vérifier que la macro va s'exécuter (vérifiez le).

Définir la valeur d'un champ dans une macro

On peut, grâce à une macro, modifier le contenu d'un champ d'une table. Imaginons que nous voulions que le nom de chaque client saisi soit en majuscules (on pourrait très bien le faire à partir du masque de saisie), nous allons pour cela utiliser une macro qui va modifier le contenu d'un champ lors d'un événement.

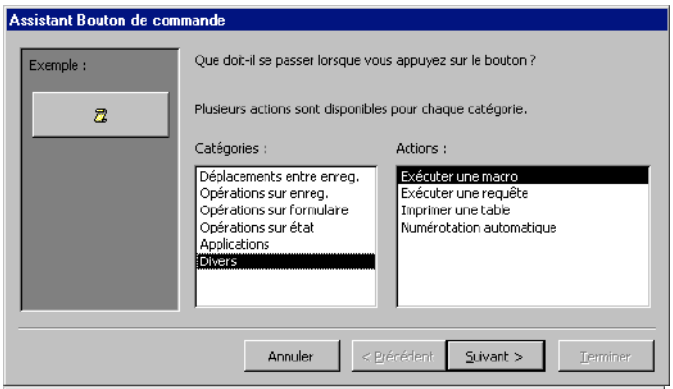
☞ Créez une nouvelle macro et utilisez l'action « *DéfinirValeur* ». Cette action prend deux paramètres : *Elément* et *Expression*. *Elément* indique ce qui est à modifier (ici le champ *AuNom*) et *Expression* indique la nouvelle valeur du champ. Vous devrez trouver vous-même quelle fonction utiliser pour changer en majuscules la chaîne de caractère correspondant au nom de l'auteur.

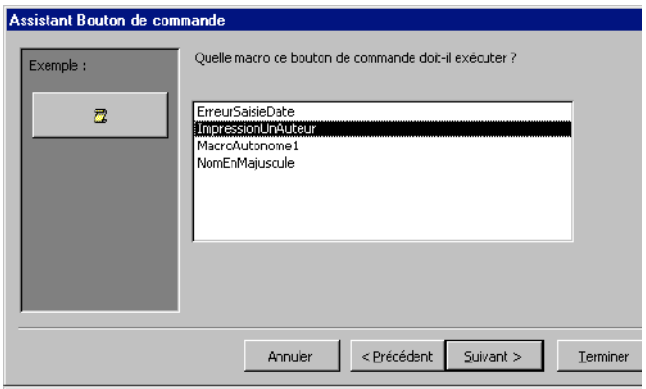
☞ Sauvegardez la macro sous le nom « *NomEnMajuscules* » et associez la à l'événement « *SurSortie* » du contrôle de nom du formulaire. Essayez de modifier le nom d'un des auteurs lorsque vous quittez le contrôle, le nom doit se changer automatiquement en majuscules.

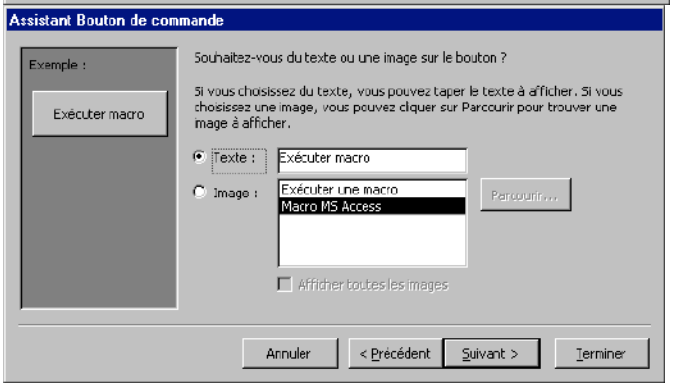
Mettre un bouton dans un formulaire

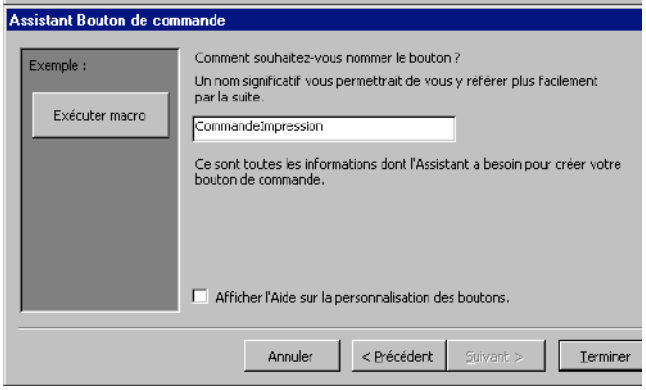
Un bouton de commande peut être placé sur un formulaire, il permet d'exécuter n'importe quelle action d'**ACCESS**. Dans le formulaire « *Formulaire Auteurs* », nous allons ajouter un bouton qui déclenchera l'impression du formulaire associé à l'auteur courant.

☞ Vous allez procéder en trois étapes. D'abord, on ajoute l'action « *ExécuterCommande* » qui permet d'exécuter toutes les opérations possibles dans **ACCESS**, on choisit l'action « *SélectionnerEnregistrement* » qui va sélectionner l'enregistrement courant. Ensuite on ajoute l'action « *Imprimer* » en choisissant *sélection* parmi la définition de ce qui doit être imprimé. Enfin on ajoute l'action « *AtteindreContrôle* » qui va positionner le curseur sur le contrôle *AuID*. Sauvegardez la macro sous le nom « *ImpressionUnAuteur* ».
☞ Insérez ensuite un bouton de commandes dans le formulaire « *Formulaire Auteurs* » en choisissant parmi les actions possibles, les macros dans la catégorie *Divers*.









☞ Cherchez dans les propriétés du bouton, à quel événement a été affectée la macro, et ajoutez la pour l'événement de double-clic.

Application professionnelle

Afin de donner une allure plus professionnelle à notre base de données, nous allons créer un formulaire qui contiendra des boutons liés à des commandes pouvant être des macros.
☞ Tout d'abord créez un nouveau formulaire sans utiliser l'assistant et mettez le en forme de façon à ce qu'il ressemble à celui de la page suivante (sans les boutons).
☞ Créez quatre macros servant à ouvrir respectivement les tables Auteurs, Livres, Catégories et Editeurs.
☞ Créez ensuite quatre boutons associés à ces quatre macros.
☞ Créez pour finir tous les autres boutons (voir la figure pour savoir quelle action associer à chaque bouton).
☞ Sauvegardez le formulaire sous le nom « Démarrage ».

☞ Pour que le formulaire soit lancé dès que l'on ouvre la base de données, utilisez le Menu *Outils* → *Démarrage* et sélectionnez le formulaire nommé « *Démarrage* » comme formulaire devant être affiché au chargement de la base de données.

L'algèbre relationnel

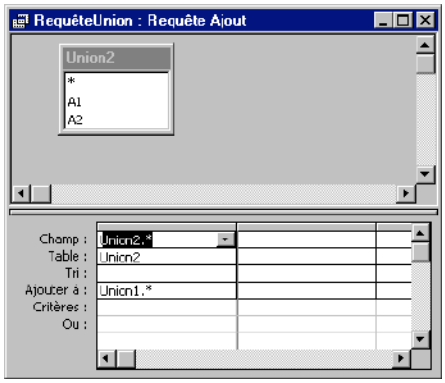
Nous allons examiner l'algèbre relationnel. Les opérations de l'algèbre relationnel sont implantées dans **ACCESS** sous la forme de requêtes. Toutes les tables sont données au paragraphe 0. Vous commencrez par créer une nouvelle base de données nommée « *AlgRel.mdb* »

L'union

Si S et T sont deux tables avec le mêmes attributs, $S \cup T$ est la table obtenue en incluant toutes les lignes de S et T.

☞ Créez les deux tables *Union1* et *Union2*.

☞ Créez une requête nommée « *RequêteUnion* » en mode création et ajoutez la table *Union2*. Transformez la requête en requête *Ajout* vers *Union1*. Glissez l'astérisque (*) du schéma de *Union2* dans la première cellule de champ. Lancez la requête et vérifiez que la table contient bien plus d'enregistrements.



L'intersection et la différence

Si *S* et *T* sont deux tables avec les mêmes attributs, $S \cap T$ est la table obtenue en ne gardant que les lignes apparaissant à la fois dans les deux tables.

☞ Créez les deux tables *Intersection1* et *Intersection2*

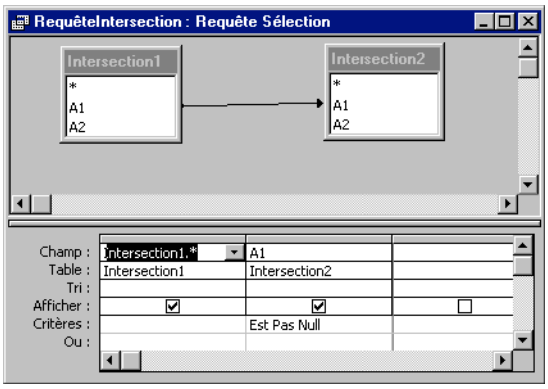
☞ Créez une requête nommée « *RequêteIntersection* » et ajoutez les deux tables. Créez une relation entre les deux champs *A1*. Cliquez avec le bouton droit sur la relation et choisissez « *Propriétés de jointure* ». Sélectionnez l'option 2 qui inclut tous les enregistrements de *Intersection1* et *Intersection2* qui ont le champ *A1* identique.

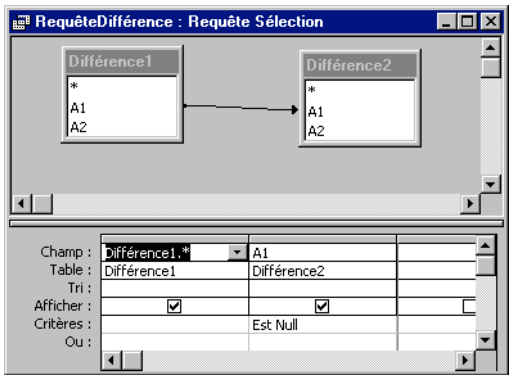
☞ Glissez l'astérisque de *Intersection1* vers la grille de conception et glissez *A1* de *Intersection2* vers la seconde colonne.

☞ Lancez la requête, les lignes où *Intersection2.A1* n'est pas NULL forment l'intersection, les lignes ou *Intersection2.A1* est NULL forment la différence.

☞ Rajoutez *Est pas Null* comme critère pour obtenir l'intersection.

☞ Créez les deux tables *Différence1* et *Différence2* et une nouvelle requête nommée « *RequêteDifférence* » identique à « *RequêteIntersection* » puis rajoutez *Est Null* pour obtenir la différence.



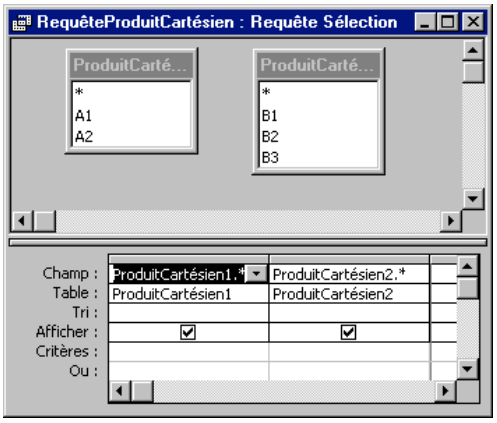


Le Produit cartésien

Si *S* et *T* sont deux tables, le produit cartésien $S \times T$ est la table où chaque enregistrement est formé par la combinaison de chaque ligne de *S* et *T*.

☞ Créez les deux tables *Produit1* et *Produit2*.

☞ Créez une nouvelle requête nommée « *RequêteProduitCartésien*» utilisant les deux précédentes tables. Glissez les astérisques de chaque schéma vers la grille de conception. Lancez la requête pour obtenir le produit cartésien.

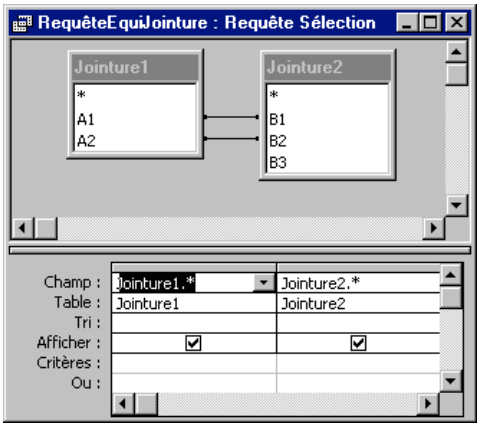


La Jointure

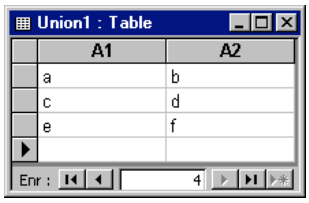
Faire la jointure de deux tables revient à combiner les lignes de deux tables sur la base d'une comparaison (d'égalité pour l'équi-jointure) dans des colonnes sélectionnées.

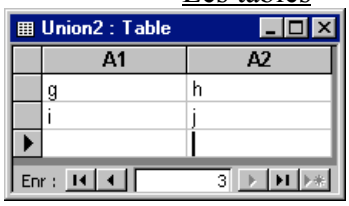
☞ Créez les deux tables *Jointure1* et *Jointure2*.

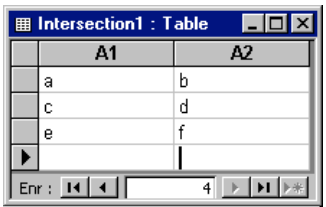
☞ Créez une nouvelle requête nommée « *RequêteEquiJointure*» utilisant les deux précédentes tables. Nous volons créer la jointure basée sur l'égalité entre *A1* et *B1* ainsi qu'entre *A2* et *B2*. Glissez le nom du champ *A1* sur *B1* et *A2* vers *B2*. Glissez les astérisques vers la grille de conception. Lancez la requête pour obtenir une equi-jointure.

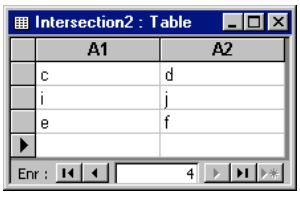


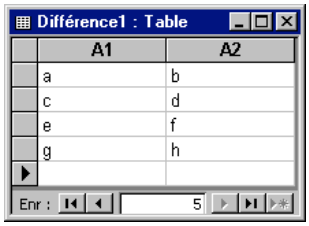
Les tables

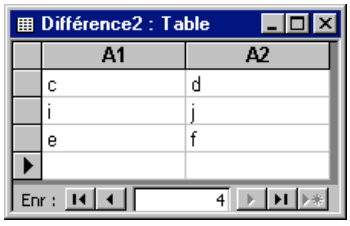


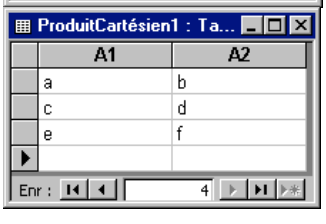












ProduitCartésien2 : Table

B1	B2	B3
g	h	i
j	k	l

Enr : 3 sur 3

Jointure1 : Table

A1	A2
a	b
c	d
e	f

Enr : 4

Jointure2 : Table

B1	B2	B3
g	h	i
j	k	l
c	d	x
c	d	y
c	y	z

Enr : 6 sur 6

Exercice récapitulatif

Vous allez maintenant refaire toutes le démarches nécessaires à la création d'une base de données relationnelle. Vous disposez du schéma relationnel suivant :
ETUDIANTS(NumEtud, NomEtud, PrenomEtud, AnneeStage, *NumEnt*)
ENTREPRISES(NumEnt, NomEnt, NomResp, PrenomResp, TelResp)

☞ Créez une nouvelle base de données nommée « *StageIUT.mdb* », créez les tables, spécifiez les format de données ainsi que les masques de données, précisez les clés primaires.
☞ Créez l'association entre les deux tables en respectant l'intégrité référentielle.
☞ Construisez une requête permettant de trouver :

- Le nom de l'entreprise dans laquelle un étudiant a effectué son stage (le nom de l'étudiant devra être saisi dans une boîte de dialogue, puis le prénom dans une autre),
- Le nom et le prénom du responsable de stage,
- Son numéro de téléphone.

☞ Construisez une requête permettant de trouver tous les étudiants ayant fait leur stage dans une entreprise (le nom de l'entreprise devra être saisi dans une boite de dialogue). Cette requête doit donner le nom et le prénom des étudiants ainsi que l'année du stage.
☞ Faites-en ensuite un état.

☞ Construisez un état donnant les références de toutes les entreprises qui ont pris un ou des stagiaire(s). Les noms et prénoms des stagiaires doivent apparaître sous chaque entreprise ainsi que l'année du stage.

☞ Créez une requête donnant le nombre d'étudiants pris en stage par chaque entreprise.

☞ Créez un formulaire de démarrage pour l'application, vous intégrerez des boutons pour effectuer chacune des requêtes ainsi qu'un bouton permettant de sortir d'**ACCESS**.

Etudiants : Table

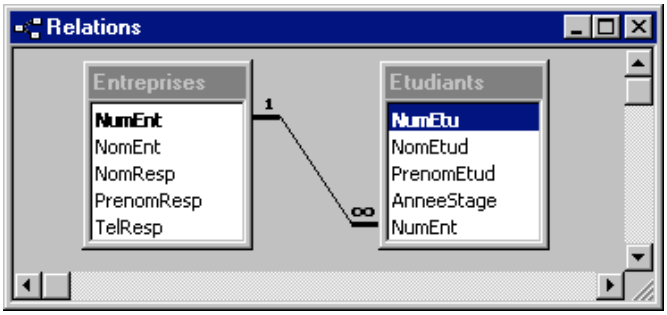
NumEtu	NomEtud	PrenomEtud	AnneeStage	NumEnt
2	Lhermitte	Thierry	2001	2
3	Villeret	Jacques	1999	1
4	Huster	Francis	2000	3
5	Frot	Catherine	1998	1
6	Prevost	Daniel	1999	2
7	Depardieu	Gérard	2000	3
8	Auteuil	Daniel	2001	2
9	Beart	Emmanuelle	2001	1
(NuméroAuto)			2001	0

Enr : 9 sur 9

Entreprises : Table

NumEnt	NomEnt	NomResp	PrenomResp	TelResp
1	Lanesra	Bourvil	André	02-33-21-22-23
2	Mageco	De Funes	Luis	02-33-24-25-26
3	Telcaal	Raynaud	Fernand	02-33-27-28-29
(NuméroAuto)				00-00-00-00-00

Enr : 4 sur 4

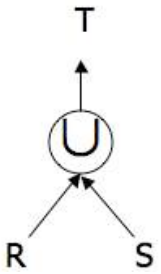


4. Algèbre relationnel

- Une requête sur BDD fournit une nouvelle table en résultat
- Langage de requêtes algébrique
- Ensemble d'opérateurs qui s 'appliquent aux relations
- Résultat : une nouvelle relation qui peut à sont tour être manipulée

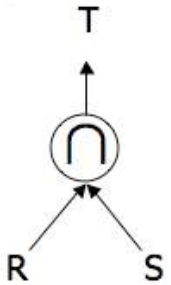
4.1. L'union

- Union : T = R ∪ S ou T=UNION(R,S) et R et S doivent avoir le même schéma
- Ex: R et S sont les relations PRODUIT de deux sociétés qui fusionnent et veulent unifier leur catalogue
- Notation graphique :



4.2. L'intersection

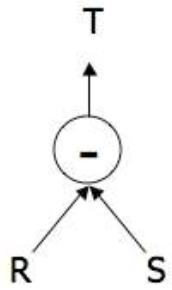
- Intersection : T = R ∩ S ou T=INTERSECT(R,S) et R et S doivent avoir le même schéma
- Ex: Permet de trouver les produits communs aux catalogues de deux sociétés
- Notation graphique :



4.3. La différence

- Différence : T = R - S ou T=MINUS(R,S) et R et S doivent avoir le même schéma
- Ex: Permet de retirer les produits de la relations S existants dans la relation R

■Notation graphique :

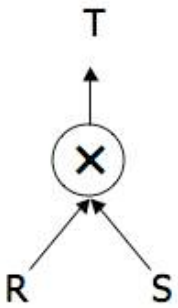


4.4. Le produit cartésien

■Produit cartésien : T = R × S ou T=PRODUCT(R,S) et R et S peuvent ne pas avoir le même schéma

■Associe chaque n-uplet de R à chaque n-uplet de S

■Notation graphique :



4.5. Exemple de produit cartésien :

NumProd	Dési
0	P1
1	P2

×

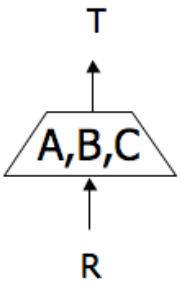
NumFour	RaisonSoc
10	F1
20	F2
30	F3

=

NumProd	Dési	NumFour	RaisonSoc
0	P1	10	F1
1	P2	10	F1
0	P1	20	F2
1	P2	20	F2
0	P1	30	F3
1	P2	30	F3

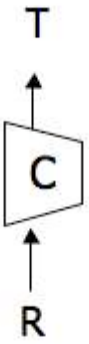
4.6. La projection

- Projection : T = Π<A,B,C>(R) ou T=PROJECT(R/A,B,C)
- T ne contient que les attributs A,B et C de R
- Ex: Noms et prénoms des clients
- Notation graphique :



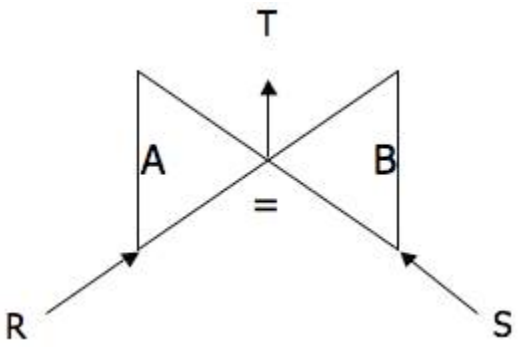
4.7. La restriction

- Restriction : T = σ<C>(R) ou T=RESTRICT(R/C)
- T ne contient que les attributs de R qui satisfont la condition C
- Ex: C=Clients qui habitent Lyon
- Notation graphique :

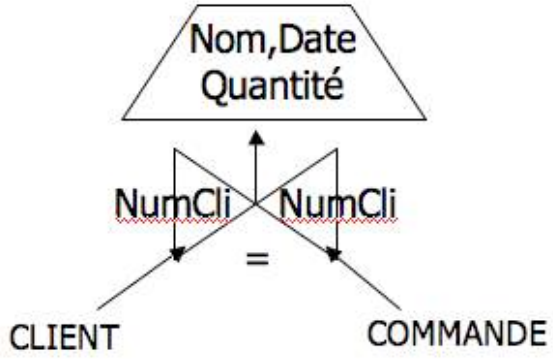


4.8. La jointure

- Jointure naturelle : T = R S ou T=JOIN(R,S)
- Produit cartésien R × S avec la restriction A=B sur les attributs A∈R et B ∈S
- Notation graphique :



4.9. Exemple de jointure



- Ex: commande avec le nom du client et pas seulement son numéro
- Nb: Requête = enchaînement d'opérations

4.10. Décomposition d'une relation

- Une décomposition est le remplacement d'une relation R par une collection de relations $R_1,R_2,\ldots,R_n$ obtenues par des projections de R et telle que la relation résultant des jointures

$$R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$$

constitue un schéma relationnel équivalent à R.

4.11. Préservation du contenu

- Les n-uplets obtenus par jointure doivent être exactement ceux de R: la recomposition par jointure ne doit ni enlever ni ajouter des n-uplets
- Théorème : Une décomposition $R=\{R_1,R_2\}$ préserve le contenu ssi

$R_1 \cap R_2 \rightarrow R_1 - R_2$
$R_1 \cup R_2 \rightarrow R_2 - R_1$

4.12. Préservation des dépendances

■Un algorithme de décomposition préserve les dépendances si les dépendances initiales de D peuvent être reconstruites à partir de D_i. Autrement dit si la fermeture de D est identique à la fermeture de l'union des D_i :

$$(D_1 \cup D_2 \cup \ldots \cup D_n)^+ = D^+$$