الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

UNIVERSITE BADJI MOKHTAR - ANNABA
BADJI MOKHTAR – ANNABA UNIVERSITY
Faculté de Technologie
Département d'Informatique

جامعة باجي مختار – عنابـــــــــــة

Support de cours

À l'usage des étudiants de 3ème Année Licence en Mathématique Appliquées

INTRODUCTION AUX SYSTEMES D'INFORMATION ET AUX BASES DE DONNEES

Par : **Dr Lamia MAHNANE**

Année Universitaire : 2021/2022

Préambule

Ce support de cours intitulé « Introduction aux systèmes d'information et aux bases de données » est destiné aux étudiants du semestre S5 de la troisième année de licence département de Mathématique de l'Université Badji Mokhtar, Annaba. Il a été enseigné au département à partir de 2013 jusqu'à 2022. Le format de ce cours est composé de deux séances de 2h de cours associées à deux séances de travaux pratiques accessible sur le lien : https://elearning-facsc.univ-annaba.dz/enrol/index.php?id=91

Ce cours permet d'acquérir les éléments de base sur les systèmes d'information et gestion de données.

Les connaissances et les compétences visées à travers ce cours sont les suivantes :

- ✓ Qu'est ce qu'une base de donnée ?
- ✓ Quand est ce qu'on utilise les bases de données ?
- ✓ Qu'est ce qu'un Système de Gestion de Bases de Données (SGBD) ?
- ✓ Le processus de développement d'une base de données.
- ✓ Savoir élaborer un modèle entité-association.
- ✓ Savoir passer d'un modèle entité-association à un modèle relationnel.
- ✓ Savoir écrire des requêtes SQL.
- ✓ Savoir implémenter le modèle relationnel en utilisant un SGBD cible.
- ✓ Quels sont les étapes à suivre afin d'élaborer un modèle conceptuel de données.

Pour que nos objectifs soient atteints, nous avons synthétisé les informations les plus pertinentes en s'appuyant sur des sources variées (notes de cours, ouvrages, sites internet, etc.) tout en respectant le canevas officiel défini par le ministère de l'enseignement supérieur et de la recherche scientifique.

Annaba, 01 Janvier 2022

Dr. MAHNANE Lamia

Table des matières

Chapitre 3 : Le Modéle Logique de Données (MLD)

Chapitre 4 : Algèbre relationnelle

Chapitre 5 : Le langage SQL

Chapitre 6 : Application avec MS-Access

Chapitre 7 : Exercices et travaux pratiques

Liste des Figures

Chapitre 1: Système d'information et Base de Données

1. Système d'information

1.1 Introduction

Né dans les domaines de l'informatique et des télécommunications, le concept de système d'information s'applique maintenant à l'ensemble des organisations. Le système d'information coordonne grâce à l'information et les activités de l'organisation et lui permet ainsi d'atteindre ses objectifs, il est le véhicule de la communication dans l'organisation. De plus, le système d'information représente l'ensemble des ressources (les hommes, le matériel, les logiciels) organisées pour : collecter, stocker, traiter et communiquer les informations.

Un système d'information représente l'ensemble des éléments participant à la gestion, au traitement et à la diffusion de l'information au sein de l'organisation.

1.2 Notion de système

Un système est un ensemble d'éléments rassemblés pour réaliser un objectif : produire des sorties par transformation d'un ensemble d'entrées. Une entreprise par exemple est un système (composé d'hommes, de matériels, de méthodes….) qui transforme de la matière première en produit finis. Un système peut être représenté par le schéma suivant :

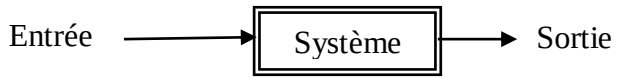


Figure 1.1 : Représentation d'un système

1.3 Système d'entreprise

Un système d'entreprise est composé de trois systèmes : Le système opérant qui constitue la machine proprement dite de production et de transformation des entrées en produits finis, le système de pilotage appelé aussi système de gestion qui pilote l'organisation et constitue son cerveau pensant et enfin le système d'information.

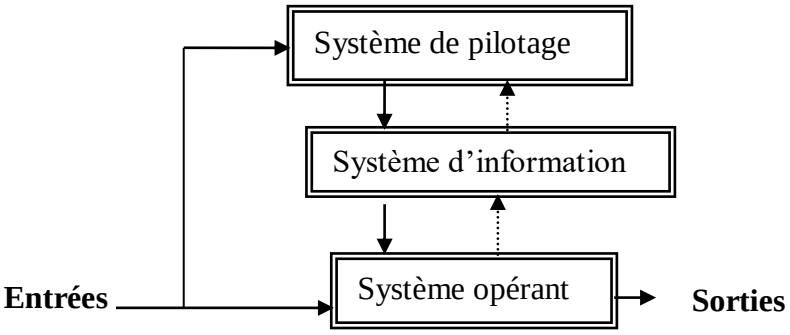


Figure 1.2 : La composante d'un systme d'information

- ***Le système opérant*** : C'est ce qui est à la base de toute organisation, c'est ce système qui permet la transformation de l'information dont l'objectif est de la restituer à la bonne personne, il correspond aux différents services d'une entreprise.

  Exemple : Si l'on considère une université (qui est un exemple typique d'une organisation), le système opérant est constitué des services et des départements qui organisent le concours d'entrée, les délibérations et les jurys, les cours, les examens, les corrections de copies. Les élèves qui suivent les cours, qui passent des examens … etc. font aussi partie de ce système opérant. C'est l'obtention du diplôme après cinq ans d'étude qui est -entre autre -le produit final du système opérant à l'université. Le flux physique est donc constitué des élèves, des concours, des cours, des examens, des résultats scolaires, des rapports publiés, … etc.

- ***Le système de pilotage*** : c'est ce qui va contrôler et piloter le système opérant, il se situe donc à la tète du système d'information fixant les objectifs et prenant les décisions.

  Toute organisation est pilotée par une direction, une équipe dirigeante.

  Ce système de pilotage a pour mission de conduire l'organisation vers des objectifs qui lui sont fixés, et de vérifier que ces objectifs ont bien été atteints. Ce qui nécessite souvent un contrôle continu du fonctionnement du système opérant et d'éventuelle modifications (recrutement, investissement, nouveaux développements, … etc.) à apporter au système opérant. Parallèlement donc au flux physique, il y a un flux de décision. Ce

flux correspond aux décisions prises par la direction de l'organisation pour que celle-ci fonctionne dans les meilleures conditions et puisse atteindre ses objectifs. Et toute organisation est soumise à des contraintes extérieures et intérieures qui contraignent son action et l'empêche d'évoluer librement.

En considérant toujours la même université, les décisions prises par le système de pilotage concernent le nombre d'élèves qui seront admis chaque année à l'université, le seuil d'admissibilité, la définition des programmes des études, la définition des modalités d'examen et le règlement des études, les développements futures de l'université tels que l'augmentation du nombre d'élèves création de nouvelles filières, les ouvertures de postes de recrutement, les investissements en matériel et en logiciel, … etc.

- ***Le système d'information*** : C'est ce qui intervient entre les deux autres systèmes. Ce système s'occupe de collecter, stocker, transformer et diffuser des données et informations dans le système opérant et de pilotage.

  Dans le système opérant, cette information va permettre à celui-ci de fonctionner. Car chaque individu et chaque tâche ont besoin d'être informés sur le flux physique qui la traverse.

  En général, cette information est très détaillée, ne concerne qu'un petit élément de l'organisation, et elle est tournée vers le présent.

  Dans le système de pilotage, l'information va permettre à celui-ci de prendre les bonnes décisions en étant constamment informé de ce qui se passe dans le système opérationnel.

  Cette information a tendance à être très synthétique, elle concerne une grande partie de l'organisation (si ce n'est toute l'organisation, tel que le chiffre d'affaire annuel), et elle est tournée vers le passé et/ou le futur.

  La tâche principale du SI est donc de fournir un flux d'information qui d'une part, reflète le plus fidèlement possible le flux physique, et d'autre part fournit au système opérationnel les éléments nécessaires pour son fonctionnement quotidien et au système de pilotage les éléments nécessaires à une prise correcte de décision.

Ainsi, le flux d'information est une image du flux physique. Il représente sous une forme plus ou moins réduite, tous les événements survenus dans le système opérant ainsi que tous les éléments d'information qui permettent de traiter ces événements.

Cette image est forcément une réduction de la réalité, elle ne concerne que les aspects pertinents ayant une incidence et/ou un rôle dans le fonctionnement de l'organisation.

En reprenant l'exemple de l'université, on trouvera dans son SI toutes les informations sur les élèves qui y sont inscrits : nom prénom, adresse des parents, date de naissance, parcours scolaire précédent, n° sécurité sociale, … etc. mais on ne trouvera pas ni la couleur des yeux ni le groupe sanguin (qui est une information non pertinente pour le fonctionnement de l'université).

Plus précisément, on dit que dans le SI il y a des modèles de la réalité organisationnelle.

Ces modèles ont été construits par ceux qui mettent en place le SI. La validité et la pertinence de ces modèles sont indispensables au fonctionnement du SI lui même, et elles garantissent la qualité de l'information fournie.

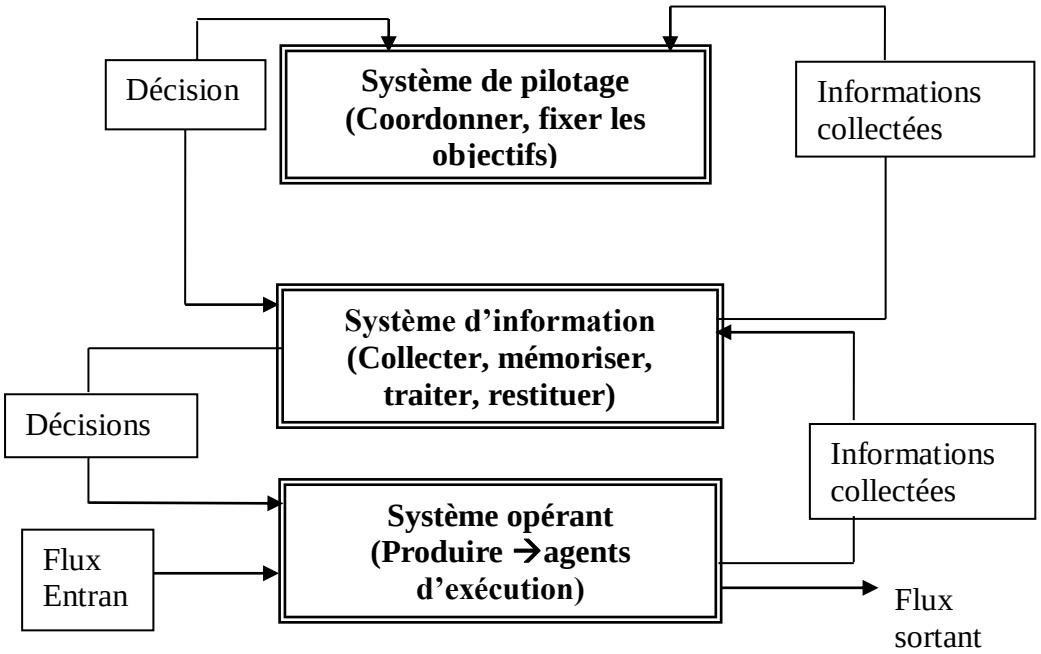


Figure 1.3 : Architecture d'un système d'information

1.4. Le fonctionnement d'un système d'information

Le système d'information réalise alors quatre fonctions essentielles :

- Collecter les informations provenant des autres éléments du système ou de l'environnement extérieur du système.
- Mémoriser les données collectées par le système
- Traiter les données stockées par le système
- Transmettre les informations vers les autres composants du système.

  La figure ci-dessous résume les 5 fonctions de base d'un système d'information. Remarquez que la communication s'effectue autant avec les systèmes de pilotage qu'avec le système opérant.

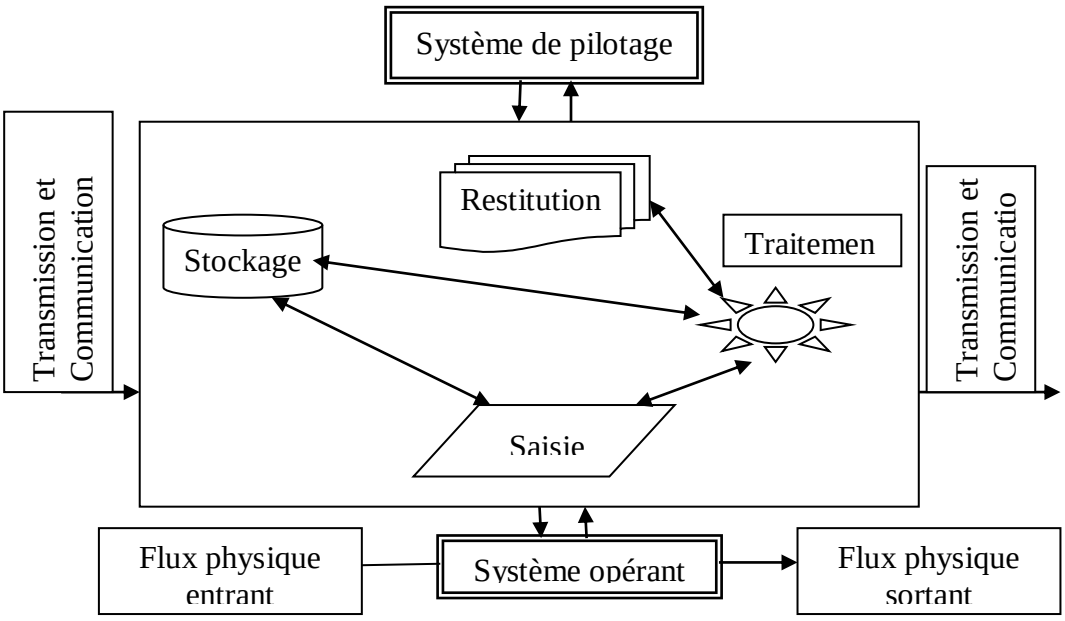


Figure 1.4 : Les cinq fonctions de base d'un système d'information

1.5 Rôle d'un système d'information dans la performance d'une entreprise

Le système d'information a deux finalités : Fonctionnelle et Sociale :

- Finalité Fonctionnelle : Le système d'information est un outil de communication entre les différents services d'une entreprise et a un rôle opérationnel et stratégique.
- Finalité Sociale : Il permet l'intégration des salariés dans la culture de l'entreprise en favorisant la vie sociale pour la diffusion de l'information.

2. Base De Données (BDD)

2.1. Définition

Une base de données est une entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondance possible. Autrement dit, elle permet de mettre des données à la disposition d'utilisateurs pour une consultation, une saisie ou bien une mise à jour. Il est possible d'accéder à la base de données par plusieurs utilisateurs simultanément. Une base de données peut être locale ou bien répartie où les données sont stockées sur des machines distantes.

Nous donnons quelques exemples :

- **Organisation** : Une bibliothèque

  Données : Les livres, les emprunts, les emprunteurs

- **Organisation** : Une Université

  Données : Les étudiants, les enseignants, les cours, etc.

2.2. Caractéristiques

Une base de données répond généralement à trois critères suivants :

- L'exhaustivité : Implique la présence dans la base des données, de tous les renseignements qui ont trait aux applications en question.
- La non-redondance : Implique la présence d'un renseignement donné une et une seule fois.
- La structure : Implique l'adaptation du mode de stockage des renseignements aux traitements qui les exploiteront et les mettront à jour, ainsi qu'au coût de stockage dans l'ordinateur.

2.3. Rôles des Base de Données (BDD)

Les bases de données ont les rôles suivant :

- Décrire les données qui seront stockées ;
- Manipuler ces données (Mise à jour des informations) ;
- Consulter les données et traiter les informations obtenues (sélectionner, trier, calculer, agréger,...) ;
- Définir des contraintes d'intégrité sur les données (contraintes de domaines, d'existence,.) ;
- Assurer les protections d'accès (mots de passe, autorisations d'accès...) ;

- Résoudre les problèmes d'accès multiples aux données (blocages, inter blocages) ;
- Prévoir des procédures de reprise en cas d'incident (sauvegardes, journaux,...).

3. Systèmes de Gestion de Base de Données (SGBD)

3.1 Définition

La gestion de la base de données se fait grâce à un système appelé SGBD (Système de Gestion de Bases de Données) ou en anglais DBMS (DataBase Management System).

Le SGBD est un ensemble de services (applications logicielles) permettant la manipulation des bases de données. Un système de gestion de base de données peut être défini comme un ensemble de logiciels prenant en charge la structuration, le stockage, la mise à jour et la maintenance des données. Autrement dit, il permet de décrire, modifier, interroger et administrer les données. C'est, en fait, l'interface entre la base de données et les utilisateurs.

Exemples: MySQL, PostgreSQL, Oracle, Microsoft SQLServer, etc.

Dans ce cours, on va utiliser un outil de bureautique de gestion de bases de données MS-ACCESS.

3.2. Objectifs des SGBD

L'objectif essentiel d'un SGBD est de garantir l'indépendance des données par rapport aux programmes (La possibilité de modifier les schémas conceptuel et interne des données sans modifier les programmes par exemple). Le but est d'éviter une maintenance coûteuse des programmes lors des modifications des structures logiques et physiques des données. Par cela, on trouve l'indépendance **physique et logique.**

En plus, pour assurer une meilleure indépendance des programmes aux données il est important de **manipuler** les données aussi bien en interrogation qu'on mise à jour via des langages. De ce fait, l'*accèsaux données* reste invisible aux programmes d'application. Les descriptions de données sont établies par les administrateurs des données, donc, le SGBD doit faciliter cette tâche.

Lorsque le SGBD met en commun les données d'une entreprise dans une BDD, et que plusieurs utilisateurs accèdent simultanément aux données, il est nécessaire de garantir : *l'efficacité des accès, le partage des données, la protection* de la BDD contre les mises à jour erronées ou non autorisées et la *sécurité* des données en cas de panne.

On peut donc résumé les objectifs des SGBD comme suit:

- **Indépendance physique** : Possibilité de modifier l'organisation physique (accès) sans modifier les programmes (modifier l'organisation physique des fichiers, ajouter ou supprimer des méthodes d'accès) sans changer le schéma conceptuel. Cela présente deux avantages : le fait de ne pas manipuler des entités complexes rend les programmes d'application plus simples à écrire, la modification des applications n'est pas obligatoire dans le cas de modification des caractéristiques du niveau physique.
- **Indépendance logique :** Le SGBD permet de modifier le schéma conceptuel et interne des données sans changer le programme d'application. Donc le niveau conceptuel peut être modifie sans remettre en cause le niveau physique « sans modifier des schémas externes).
- **Manipulation des données :** Permettre à tous types d'utilisateurs d'accéder à la base selon leurs besoins et connaissances. Par conséquent, un ou plusieurs :

- Administrateurs : Ils ont la possibilité de décrire les données aux niveauxinterne et logique,
- Développeurs d'applications écrivent des programmes d'application pour les utilisateursfinaux ou pour eux-mêmes, cela est à partir du niveau conceptuel ou externe,

- Utilisateurs peuvent manipuler les données via un langage simple dont ils ont besoin.

- **Administration facilitée des données :**

  La centralisation des descriptions de données faites par un groupe spécialisé entraine une difficulté d'organisation. Le SGBD permet de décentraliser cette

description à travers des outils. Pour cela, un dictionnaire de données dynamique peut aider les concepteurs de BDD.

- **Redondance contrôlée des données :**

La suppression des données redondantes permet d'assurer la cohérence de l'information ainsi que la simplification des mises à jour. En effet, avec les BDD réparties, il est préférable de gérer par le système des copies multiples de données. L'objectif est d'optimiser les performances en interrogation tout en évitant les transferts sur le réseau et permettre le parallélisme des accès. Conséquemment, parfois la redondance gérée par le SGBD est nécessaire spécialement au niveau physique des données.

Par contre, il faut éliminer la redondance anarchique qui force les programmes utilisateurs à mettre à jour une même donnée plusieurs fois. Donc, il faut bien contrôler la redondance.

- **Cohérence des données :**

Dans une approche BDD, les données ne sont pas indépendantes (par exemple, il peut exister certaines dépendances entre ces données). En d'autres termes, souvent une donnée ne peut pas prendre une valeur quelconque.

Exemples : Une note doit être supérieur ou égale à 0 et ne doit pas dépasser 20, un salaire mensuel doit être supérieur à 40 000 Dinar et doit raisonnablement rester inférieur à 200 000 Dinar.

Un SGBD doit veiller à ce que ces règles soient respectées par les applications lors des modifications des données et ainsi assurer la cohérence des données. Ces règles sont appelées contraintes d'intégrité.

- **Partage des données :**

L'objectif est ici de permettre le partage des données entre différents utilisateurs et applications. Un utilisateur n'a pas à se soucier si quelqu'un d'autre travaille sur les mêmes informations au même moment et peut accéder aux données en consultation ou en mise à jour comme s'il était seul. Le système doit gérer les conflits en refusant ou en retardant éventuellement un ou plusieurs accès. Il faut assurer que le résultat d'une

exécution simultanée de transactions est le même que celui d'une exécution séquentielle. Par exemple, on ne pas autoriser la réservation du même siège pour deux passagers différents.

- **Sécurité des données :**

La sécurité des données consiste à :

- Refuser les accès aux personnes non autorisées ou mal intentionnés. Le système doit présenter un mécanisme de vérification des droits d'accès aux objets de la base.

  Par exemple : Un employé peut connaître seulement les salaires des personnes qu'il dirige mais pas le salaire des autres employés de l'entreprise.

- Protéger les données contre les pannes. Le système doit garantir des reprises aprèspanne tout en restaurant la BDD dans le dernier état cohérent avant la panne.

- **Efficacité des accès aux données :**

Offrir la possibilité aux utilisateurs de manipuler les données à partir de langages hôtes (Pascal, Fortran, C, Java...) tout en assurant une efficacité et rapidité au niveau des accès sur les supports.

Avant qu'une BDD prenne sa forme finale (par exemple, une forme utilisable par un SGBD), il faut passer parune étape de conception afin de décrire les objets de la réalité ainsi que les relations entre ces objets. Pour cela, la modélisation à travers des modèles est nécessaire.

3.3. Type d'une SGBD

On distingue plusieurs modèles de bases de données ; la différence entre ces modèles est la représentation des liens entre les données de la base. Nous essayerons de parler de modèle hiérarchique, réseau, relationnel et le modèle objet.

3.3. 1. Les SGBD hiérarchiques

Ce sont les premiers SGBD apparus (notamment avec **IMS d'IBM**). Ces systèmes de gestions de bases données font partie des SGBD navigationnelles

constituées d'une gestion de pointeurs entre les enregistrements. Le schéma de la base de données est une arborescence.

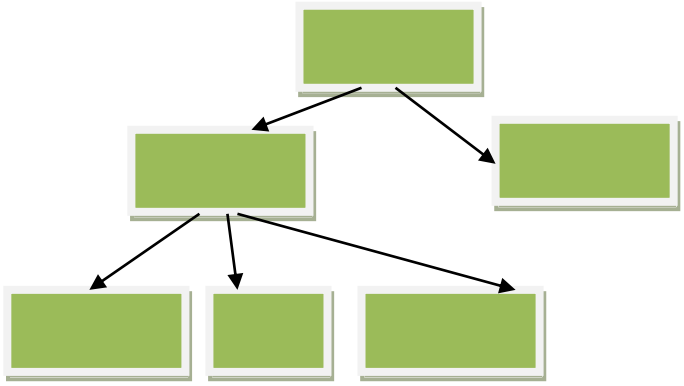


Figure 1.5: Les SGBD Hiérarchique

3.3. 2. Les SGBD réseaux

Les SGBD réseaux viennent juste après le SGBD hiérarchique, elles ont très vite détrôné le SGBD réseaux dans les années 70. Ce sont aussi des SGBD navigationnelles qui gèrent des pointeurs entre les enregistrements. Cette fois-ci le schéma de la base de données est beaucoup plus ouvert. Sans doute les SGBD réseaux sont plus rapides que SGBD hiérarchiques.

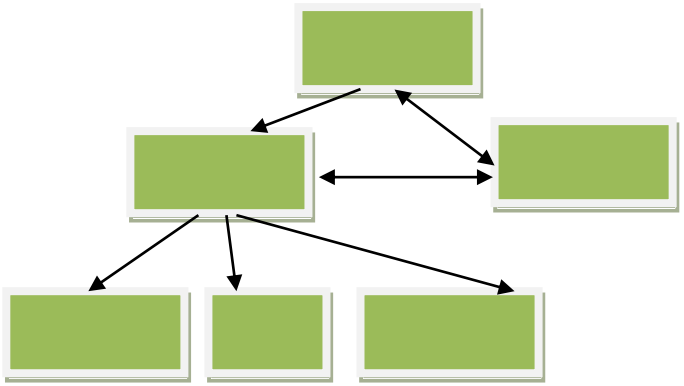


Figure 1.6 : Les SGBD Réseaux

3.3. 3. Les SGBD relationnelles

A l'heure actuelle, les SGBD relationnelles sont les plus utilisées. Les données sont représentées en des tables. Elles sont basées sur l'algèbre relationnelle et un langage déclaratif (généralement SQL).

3.3. 4. Les SGBD déductives

Dans les SGBD déductives, les données sont aussi représentées en tables, le langage d'interrogation se base sur le calcul des prédicats et la logique du premier ordre.

3.3. 5. Les SGBD objets

Les données sont représentées sous forme d'objets au sens donné par les langages orientés objet : pour simplifier, les données (au sens habituel) sont enregistrées avec les procédures et fonctions qui permettent de les manipuler. Les SGBD Orientés Objet (SGBDOO) supportent aussi la notion d'héritage entre classes d'objets. Ces dernières années le développement rapide des langages orientés objet a mis en avant les SGBDOO qui permettent la sauvegarde directe des objets manipulés par ces langages.

De plus les manipulations de données à structures complexes, en particulier dans les traitements qui font intervenir le multimédia, sont facilitées avec les SGBDOO. Le modèle relationnel a montré ses limites pour ce type de données. On verra en particulier que les structures de données complexes sont éclatées par le modèle relationnel et la reconstitution de la structure nécessite des opérations (jointures) lourdes et coûteuses en performance. Cependant les SGBDOO ne sont utilisés actuellement que pour des usages bien spécifiques et il n'existe pas encore de normes communément admises. Les SGBDR ont aussi leurs atouts. Ils reposent sur une théorie formelle plus solide que les SGBDOO. Ils sont aussi plus souples pour répondre aux interrogations de la base non prévues au départ. La recherche est active dans le domaine des SGBDOO. Pour garder leur part du marché et répondre aux besoins des utilisateurs, les grands SGBD relationnels ajoutent progressivement une couche objet à leur noyau relationnel.

Les données sont stockées sous forme d'objets, c'est-à-dire de structures appelées classes présentant des données membres. Les champs sont des instances de ces classes.

3.4 Quelques SGBD reconnus

- ✓ Oracle
- ✓ DB2
- ✓ Sybase
- ✓ SQL Server
- ✓ Ingres
- ✓ Informix
- ✓ O2
- ✓ Gemstone
- ✓ ObjectStore
- ✓ Jasmine Access, …..

4. Conception d'une base de données

La conception d'une base de données est un processus qui consiste en l'observation d'une situation réelle pour aboutir à la définition de la base de données correspondante.

Le monde informatique dispose de plusieurs méthodes de modélisation de base de données. Parmi les méthodes utilisées, nous citons la méthode MERISE.

La méthode MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise)

Est une méthode de conception, de développement et de réalisation de projets informatiques. Elle est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles.

Merise utilise une démarche de modélisation à trois niveaux. A chaque niveau correspondent à un modèle pour la représentation des données et à un modèle pour la représentation de traitements, un formalisme de représentation est associé à chaque modèle. La méthode Merise distingue 3 niveaux d'abstractions :

- ✓ Au niveau conceptuel, il s'agit de répondre aux questions : quoi ? que veut-on faire ? avec quelles données ?

On développe à ce niveau le modèle conceptuel des données (MCD) et le modèle conceptuel des traitements(MCT)

- ✓ Le niveau logique/organisationnel : permet de tenir en compte les problèmes organisationnels à partir des modèles conceptuels de données et en faisant intervenir les contraintes d'organisations, on élabore le modèle logique de données (MLD) et le modèle organisationnel des traitements(MOT).
- ✓ Le niveau Physique/Opérationnel : Le niveau physique est caractérisé par la prise en compte des contraintes technologiques : matériel, logiciel, humain, …A ce niveau, on élabore le modèle physique des données et le modèle opérationnel des traitements.

Chapitre 2 : Le Modèle Conceptuel de données (MCD)

1. Introduction

Un système d'information est défini par deux composantes : les données qui constituent l'aspect statique et les traitements qui constituent l'aspect dynamique. Merise possède l'avantage qui est d'ailleurs l'un des points clés de sa réussite, de décrire les données indépendamment des traitements. L'objectif poursuivi est la définition et l'élaboration de la structure globale des données de manière indépendante de toute contrainte organisationnelle ou technologique. La structure est appelé modèle conceptuel des donnés.

Au niveau conceptuel de la méthode on élabore pour les données, le **M**odèle **C**onceptuel des **D**onnées (MCD) et pour les traitements, le **M**odèle **C**onceptuel des **T**raitements (MCT).

On va s'intéresser dans ce cours aux modèles de représentation des données.

2. Concepts de base

2.1 Entité

Une entité est un objet concret ou abstrait qui a une existence propre, elle est aussi appelée INDIVIDU dans le formalisme individuel.

Exemple entité concrète :
Le client Mohamed
Le fournisseur Ali
L'exemplaire du livre « Programmation Matlab »
Une table, Une machine

Exemple entité abstraite :

- La matière « Anglais »
- Le service vente d'une société.

2.2 Association

Une association est une relation qui met en liaison deux ou plusieurs entités

Exemple :

- « Mohamed est marié avec Fatima », L'association mariage lie les deux entités Mohamed et Fatima.
- Amir appartient à la classe 3ieme année moyenne.
  L'association appartenance lie les deux entités Amir et Classe.

2.3. Propriété

Elle définit l'entité ou association. Autrement dit, elles apportent l'information nécessaire au système d'information.

Voici quelques exemples de propriétés :
- L'entité Client est définit par les propriétés : code client, nom, téléphone,…..
- L'entité Commande est définit par les propriétés : N°commande, date commande,…
- L'entité Facture est définit par les propriétés : N° Facture, Montant_facture,….

2.4. Identifiant Entité

L'identifiant est la propriété particulière pour chaque entité. Il permet de désigner chaque occurrence de manière unique.

Exemples :
Le matricule est un identifiant de l'entité « Voiture ». Autrement dit, ce n'est pas possible d'avoir deux voitures avec le même matricule.

3 Représentation schématique

Nous présentons dans ce qui suit comment représenté schématiquement un modèle selon la méthode de MERISE.

3.1 Schématisation (Entité-Association)

Une entité est représentée par un rectangle divisé en deux parties, une partie supérieure contient le nom de l'entité et la partie inférieure porte la liste des propriétés.
L'identifiant est repéré dans la liste des propriétés de la manière suivante :

- Il figure en première position dans la liste des propriétés.
- Il est souligné.

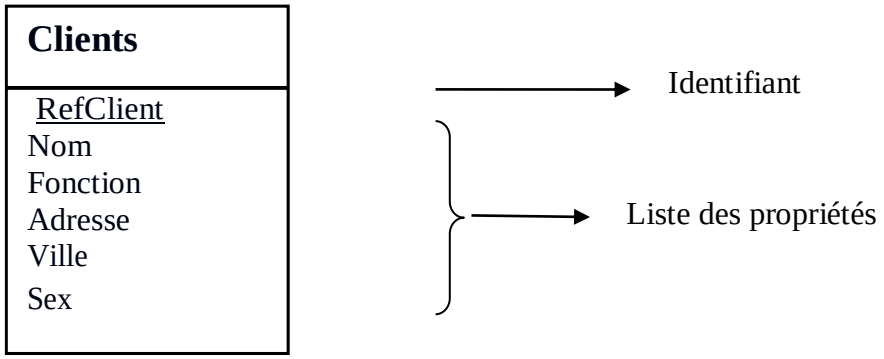


Figure 2.1 : Représentation schématique (Entité-Association)

L'association est schématisée par une ellipse. On trouve comme dans la représentation schématique d'entités deux parties, une partie supérieure inscrit le nom de l'association et dans l'autre partie, la liste des propriétés.

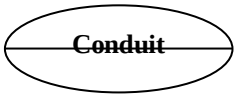


Figure 2.2 : Représentation schématique (Association)

Exemples :
La figure 2.3 représente deux entités Etudiant et Classe liées par une association d'appartenance. On peut facilement lire qu'un étudiant appartient à une classe.

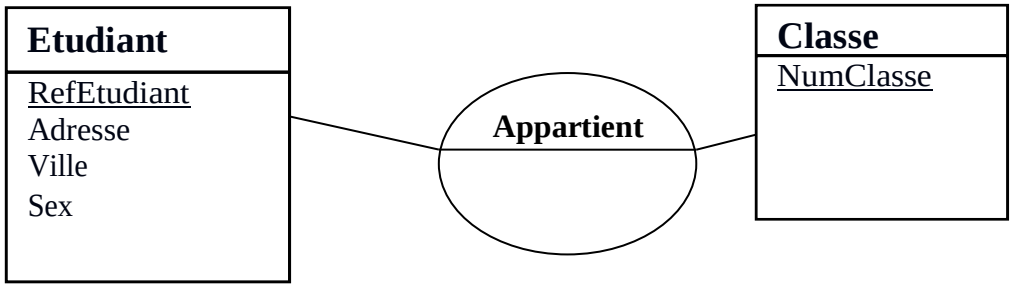


Figure 2.3 : Deux entités Etudiant et Classe liées par une association d'appartenance

En fonction du nombre d'entité relié par l'association on peut déterminer la notion d'une dimension d'une association

3.2. Dimension d'une association

C'est le nombre d'entités participant à l'association :

- Une association entre deux entités est appelée association binaire.
- Une association entre trois entités est appelée association ternaire.
- Une association entre n entités est appelée association n-n'aire.

Les figures suivantes représentent un exemple d'une association binaire et ternaire :

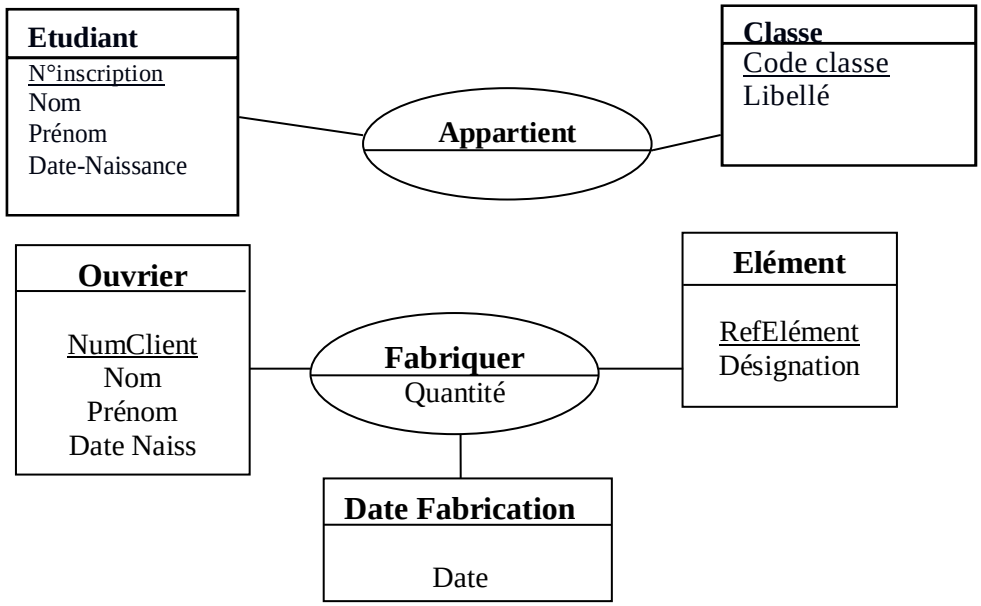


Figure 2.4 : Exemple d'une association binaire et ternaire

On passe maintenant à définir la notion de cardinalité, quelle est son rôle et qu'est ce qu'elle permet d'apporter dans ce modèle :

3.3. Les cardinalités

Les cardinalités apportent des informations importantes sur les associations dans le modèle conceptuel de données.

Prenons l'exemple précédent :

Si un étudiant peut appartenir à une ou plusieurs classes. Dans ce cas, on ne peut pas savoir si une classe contient un ou plusieurs étudiants.

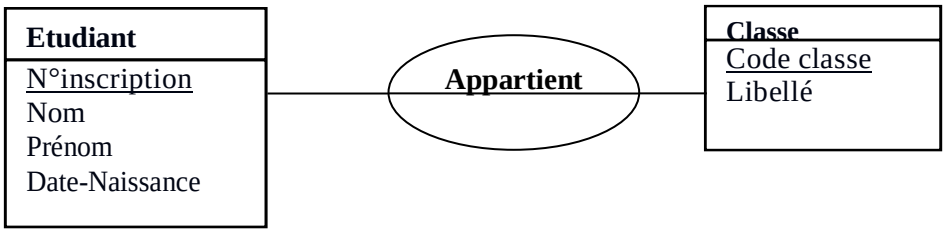


Figure 2.5 : Exemple de deux entités Etudiant et Classe

La cardinalité d'une association pour une entité constituante est constituée d'une borne minimale et d'une borne maximale :

- **Cardinalités Minimum** : nombre minimum de fois qu'une occurrence de l'entité participe aux occurrences de l'association, généralement 0, 1 ou C.
- **Cardinalités Maximum** : nombre maximum de fois qu'une occurrence de l'entité participe aux occurrences de l'association, généralement 1, N ou C.

  Autrement dit, les combinaisons des cardinalités sont des couples qui peuvent prendre comme valeur :
- (0,1 ou C) (aucun ou 1 seul), où C est une constante (C>1);
- (0, N ou C) (aucun ou plusieurs), où C est une constante (0<C<N) ;
- 1,1 (un et 1 seul)
- 0, N (au moins un ou plusieurs)

  Le premier chiffre correspond au minimum, le second correspond au maximum

Prenons l'exemple suivant :

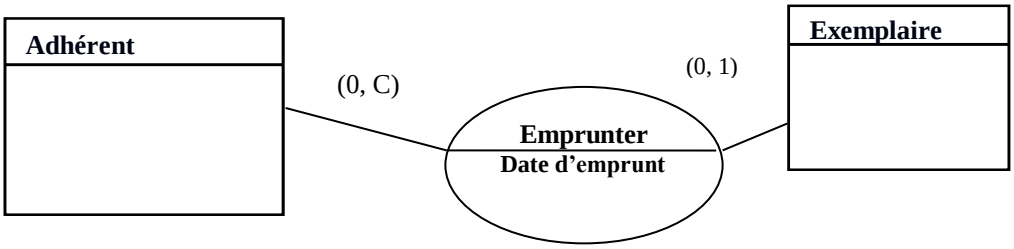


Figure 2.6 : Exemple sur les cardinalités

• La cardinalité 0,C indique qu'un adhérent peut être associé à 0, 1, 2 ou C livres, c'est à dire qu'il peut emprunter au maximun C livres.

• A l'inverse un livre peut être emprunté par un seul adhérent, ou peut ne pas être emprunté.

• Les cardinalités maximum sont nécessaires pour concevoir le schéma de la base de données
• Les cardinalités minimums sont nécessaires pour exprimer les contraintes d'intégrité

Voici des exemples avec des cardinalités différentes :

❖ **Exemple1 : Association de type (1, N) :** Un étudiant réside dans un seul et un seul département et un département peut contenir 0 ou plusieurs étudiants.

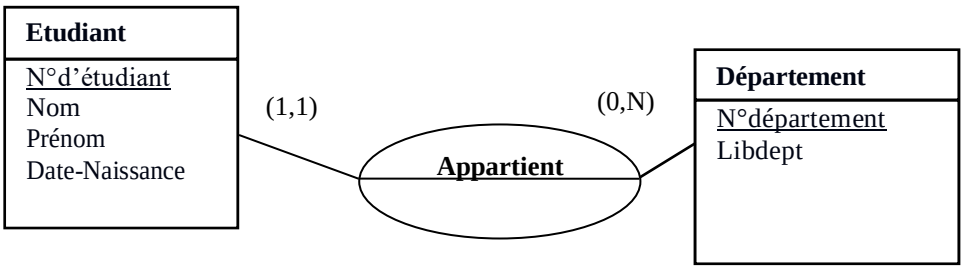


Figure 2.7 : Exemple association de type (1, N)

❖ **Exemple2 : Association de type (N, N) :** Un étudiant pratique un ou plusieurs sports, et un sport est pratiqué par un ou plusieurs étudiants

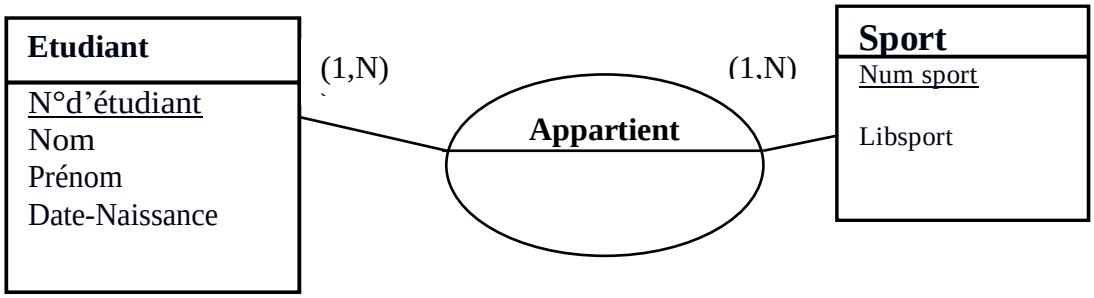


Figure 2.8 : Exemple association de type (N, N)

❖ **Association de type (N, N)** porteuse d'information **:**
Remarque : Un étudiant apprend plusieurs matières et une matière peut être enseignée à plusieurs étudiants
La propriété moyenne de l'association Noter est en relation avec les deux entités Etudiant et Matière.

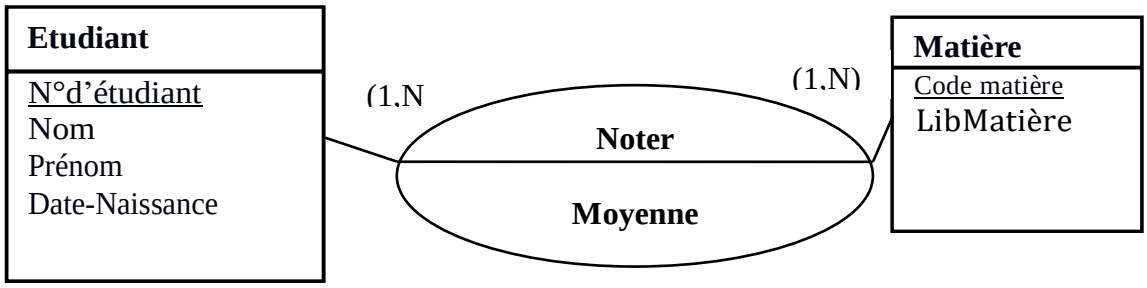


Figure 2.9 : Exemple association de type (N, N) porteuse d'information
+

❖ **Voici d'autres exemples d'association spéciale :**
Association réflexive : C'est l'association d'une entité sur même.
Exemple : Considérons l'entité-Employé d'une entreprise et l'association **conjoint.**

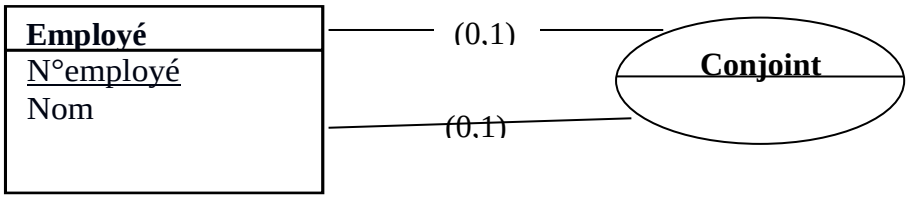


Figure 2.10 : Exemple association réflexive

(0) : Un employé peut ne pas avoir de conjoint employé dans l'entreprise
(1) : Un employé peut avoir au maximum un conjoint qui est lui-même employé dans l'entreprise.

- **Association Multiple :** Deux entités distinctes peuvent avoir plusieurs liens de dépendance. Dans ce cas elles vont être reliés par plusieurs associations .c'est ce qu'on appelle association multiple.

  Exemple :

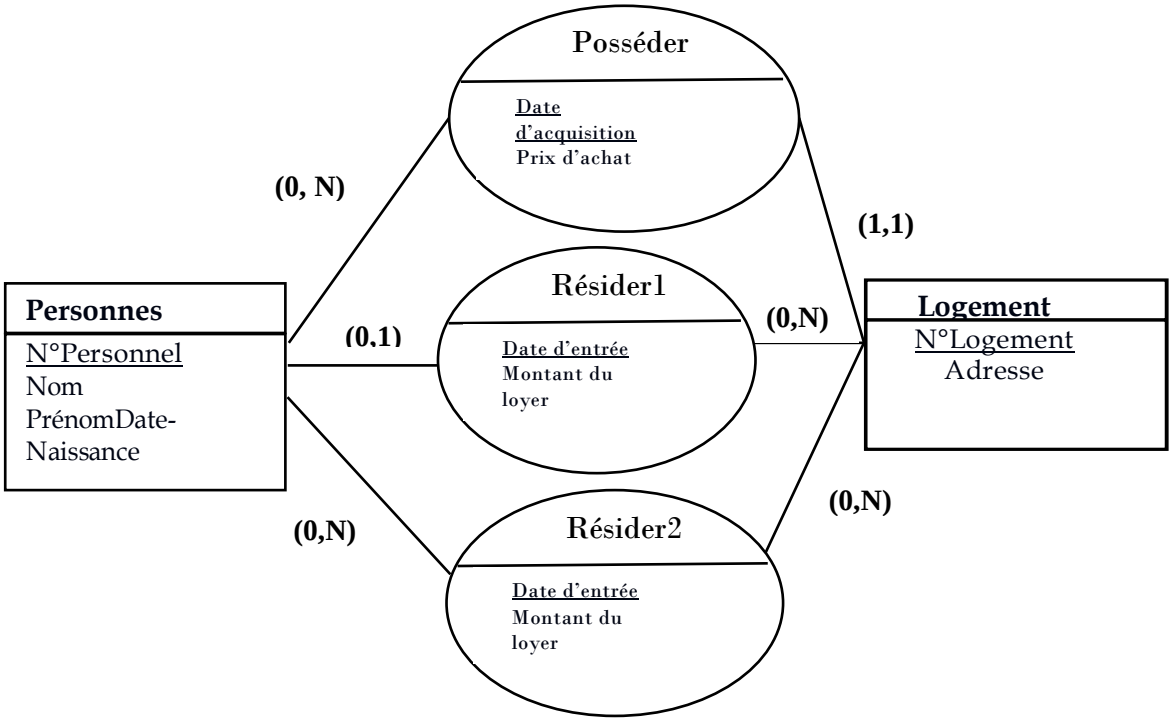


Figure 2.11 : Exemple association Multiple

Dans cet exemple issu d'une agence immobilière, une personne peut être propriétaire, résider principalement ou résider secondairement dans un logement géré par l'agence.

4. Normalisation du modèle conceptuel des données

4.1 Dépendances Fonctionnelles (DF)

Une dépendance fonctionnelle est une interrelation, un lien entre deux données ou deux groupes de données.
On distingue une source et une cible.
Définition : Pour une valeur source (partie gauche), on peut déterminer une et une seule valeur cible (Partie droite).
On dit qu'une propriété B dépend fonctionnellement d'une autre propriété A et on note :
A →B
Autrement dit à une valeur de A correspond toujours une et une seule valeur de B. la réciproque n'est pas vraie.

Exemple : NumClient →Nom

4.2. Propriétés des dépendances fonctionnelles

Les dépendances fonctionnelles ont les propriétés suivantes :

4.2.1. Union

Si on a deux DF ayant la même source, on peut les rassembler en une seule, en séparant les cibles par une virgule.
Si A →B et A→C alors on peut écrire que A→B, C
Exemple :
Référence →Désignation et Référence →prix de vente unitaire alors par union on a :
Référence →Désignation, Référence et prix de vente unitaire.

4.2.2. Transitivité

Si A →B et B→C alors on a A →C
Exemple :
NumMédecin→CodeService et
CodeService→NumHopital alors on a
NumMédecin→NumHotel

- Dépendance fonctionnelle élémentaire : C'est l'intégralité de la source qui doit déterminer la cible d'une DF.
  Si P1→P3 alors P1, P2→P3 n'est pas élémentaire.

Exemple :
N°Client, Nom →Prénom
Cette dépendance n'est pas élémentaire puisque le N°client suffit pour déterminer le prénom.
La source d'une dépendance fonctionnelle peut se composer d'une concaténation de deux ou plusieurs propriétés.
Par exemple : N°étudiant+code matière →note

4.2.3. Dépendance fonctionnelle directe

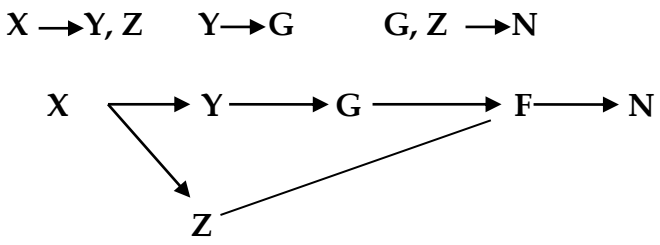
La DF ne doit pas être obtenue par transitivité. Par exemple, si P1→P2 et P2→P3 alors P1→P3 a été obtenue par transitivité et n'est donc pas directe.
A→B est directe s'il n'existe pas de propriété C : tel que A→C et C→B .
Autrement dit la dépendance fonction n'est pas le résultat d'une transitivité.

Exemple :

DF	Directe (Oui/Non)
N°Enseignant ⟶ Code Module	Oui
Code Module ⟶ Nom Module	Oui
N°Enseignant ⟶ Nom Module	Non

Les deux premières dépendances sont directes, mais la troisième ne l'est pas car elle résulte de l'application de la transitivité.
Graphe de Dépendance Fonctionnelles (GDF) : Le GDF fait apparaitre les dépendances fonctionnelles entre les données.

Exemple :



La schématisation de l'ensemble des dépendances fonctionnelles sous forme de GDF intervient à l'élaboration du modèle conceptuel des données, en respectant certaines règles.

4. 3. Normalisation

4.3.1 Première Forme Normale 1FN

Une entité est sous la 1ère Forme Normale (1FN) Si :

- L'entité possède un identifiant ;
- Tous ses attributs **sont atomiques** et non composées (Voir Figure 2.12 a) ;
- Toutes les propriétés ne reçoivent qu'une seule valeur (Monovaluées) .
- (Voir Figure 2.12 b)
- Constant dans le temps (Voir Figure 2.12 c)

Exemple :

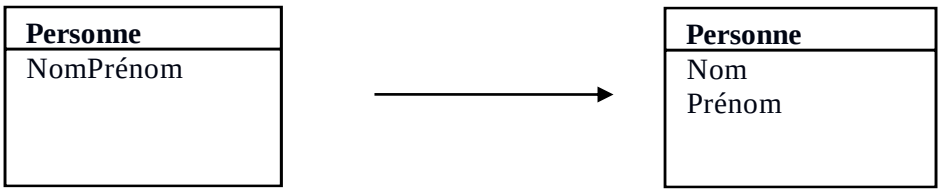


Figure 2.12a : Exemple d'un attribut composé

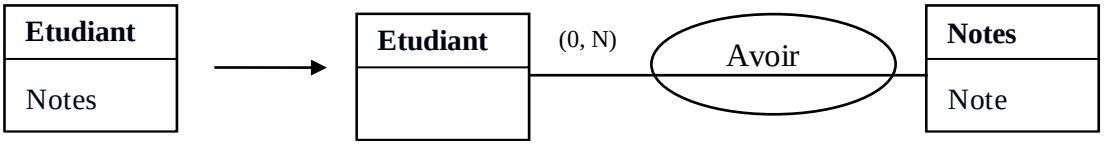


Figure 2.12b : Exemple des valeurs monovaluées

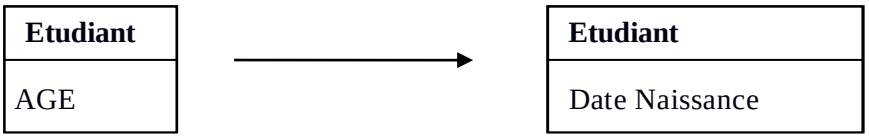
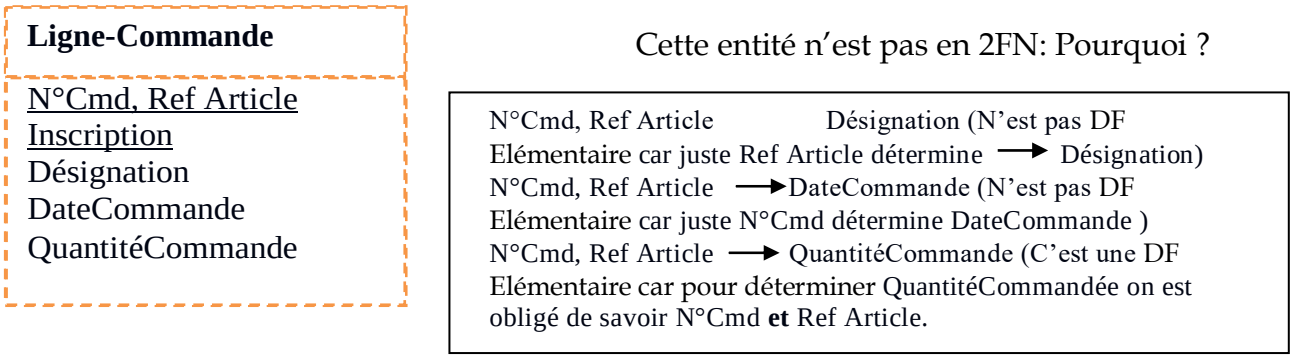


Figure 2.12c : Exemple d'un constant dans le temps

4.3.2. Deuxième forme normale 2FN

Une entité est en deuxième forme normale si elle est d'abord en 1FN et toutes les dépendances entre l'identifiant et les autres propriétés dont élémentaires :



La solution proposée est la suivante :

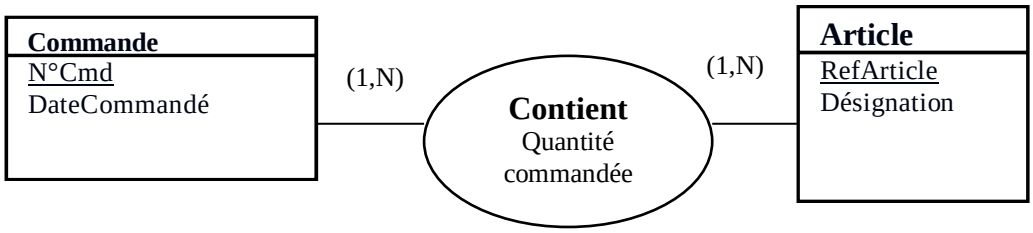


Figure 2.13 : Exemple d'une 2FN

4.3.3 Troisième forme normale 3FN

Une entité est en 3 FN si :

- Respecte 2FN ;
- Tous les attributs n'appartiennent pas à l'identifiant ne dépendent pas d'un attribut non identifiant.

Exemple :

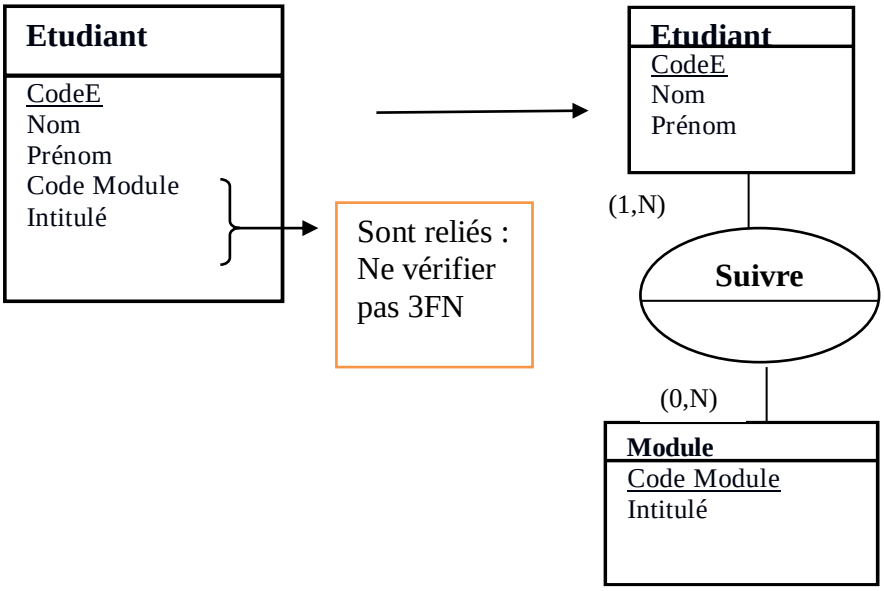


Figure 2.14 : Exemple d'une 3FN

Chapitre *3* :
Le Modèle Logique de Données (MLD)

1. Introduction

Le modèle logique de Données est un passage du Modèle Conceptuel de Données (MCD) validé vers le Modèle Physique des Données (MPD). Cette étape consiste à transformer le modèle conceptuel de données en modèle logique, selon un formalisme adapté à un type de Système de Gestion de Bases de Données (SGBD).

Pour ce faire, il existe plusieurs types de modèle : le modèle Hiérarchique, le modèle Réseau et le Modèle relationnel.

En raison des utilisations fréquentes de ce type et sa simplicité, le modèle relationnel sera traité dans ce chapitre.

2. Le modèle logique relationnel

Le modèle logique relationnel des données consiste à décrire la structure de données utilisées. Il s'agit donc de préciser la structure des données selon un modèle relationnel où les données sont enregistrées dans des tables à deux dimensions.

2.1 Relation , Tuples et attributs

- Chaque table représente une relation.
- Chaque colonne d'une table représente un attribut
- Une ligne du tableau représenté est un tuple.
- Un attribut est repéré par un nom et un domaine de définition, c'est à dire l'ensemble des valeurs qu'il peut prendre (entier, booléen , chaine de caractère)

CLIENT			
N°Client	**Nom**	**Prénom**	**Adresse**
A001	Julien	Martin	38 rue Guynemer. 75006. Paris
B002	David	Bernard	7 rue des Fleurs 37000 Tours
C003	Marie	Leroy	7 Rue rue Guynemer 36000. Toulouse

Remarque : On peut dire que la table représente l'élément de base du modèle relationnel. Elle généralement désigné par son nom.

2.2. Clé primaire

La clé primaire d'une relation est l'attribut, ou l'ensemble d'attributs, permettant de désigner de façon unique un tuple.

Prenons le même exemple précédent :

CLIENT			
N° Client	**Nom**	**Prénom**	**Adresse**
A001	Julien	Martin	38, rue Guynemer. 75006. Paris
B002	David	Bernard	7 rue des Fleurs 37000 Tours
C003	Marie	Leroy	7 Rue rue Guynemer 36000. Toulouse

Voici quelques propriétés requises de la clé primaire :

- La valeur vide (NULL) est interdite.
- La valeur de la clé primaire d'une ligne ne devrait pas changer au cours du temps.

2.3. Clé étrangère

Une clé étrangère est une clé faisant référence à une clé appartenant à une autre relation.

La transformation du modèle conceptuel des données ci-dessus en modèle logique des données permet d'obtenir le résultat suivant :

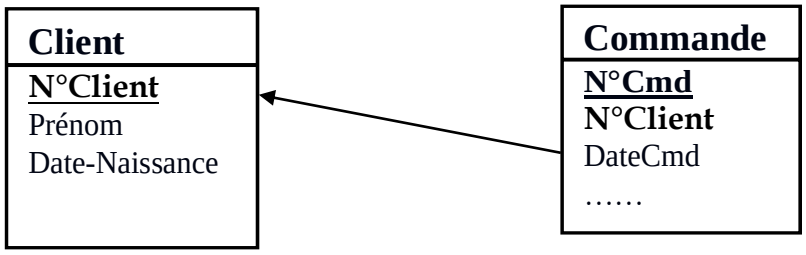


Figure 3.1 : Transformation MCD vers le MLD

Ci-dessous, un extrait de la table client et la table commande après transformation MCD vers le MLD :

CLIENT			
N°Client	Nom	Prénom	Adresse
A001	Alaoui	Mohamed	78 Rue Amirouche, Annaba
B002	Bahi	Souad	28 Rue Tarek Ben Zied, Elbouni
C003	Amirouche	Leila	7 Rue Safsaf, Sidi Hareb

COMMANDE		
N°Cmd	N°Client	Date Cmd
1	A001	10/04/2021
2	B002	25/05/2021
3	C003	09/12/2021

Dans la table « commande », les valeurs de la colonne N°Client ne doivent contenir que des valeurs prises de la colonne N°Client (identifiant) de la table « Client ».

On dit alors que la colonne N°Client de la table « commande » est une clé étrangère.

3. Passage MCD vers MLD

La traduction du MCD en modèle logique relationnel s'effectue directement par transformation des entités conceptuelles en relations, en fonction des règles de passage précises.

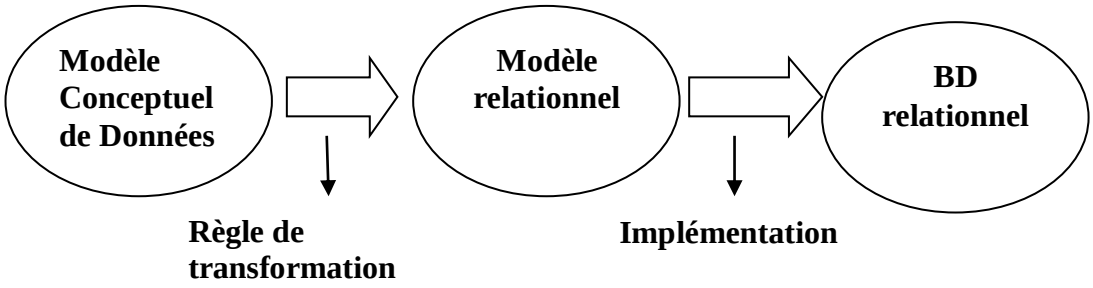


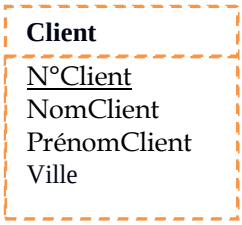
Figure 3.2 : Passage MCD vers MLD

Le passage du modèle conceptuel des données au modèle logique des données s'effectue en appliquant des règles s'appuyant sur les cardinalités des couples entités/association.

3.1. Règle de transformation

- **Règle 1** : Une entité se transforme en relation. Chaque propriété se transforme en attribut. L'identifiant de l'entité devient la clé primaire de la relation.

Exemple :



CLIENT			
N° Client	**Nom Client**	**Prénom Client**	**Ville**
1	Alaoui	Mohamed	Annaba
2	Bahi	Souad	Setif
3	Amirouche	Leila	Oran
…..	…..	……	…..

- **Règle 2** : « Association **Père-Fils** » : Une association binaire ayant des cardinalités (1,1) – (1, N) ou (1,1) – (0, N) se traduit par l'immigration de l'identifiant de l'entité père (ayant cardinalité (1, N) ou (0, N)) vers l'entité fils (cardinalité (1,1)). Nous présentons cet exemple pour mieux comprendre cette règle :

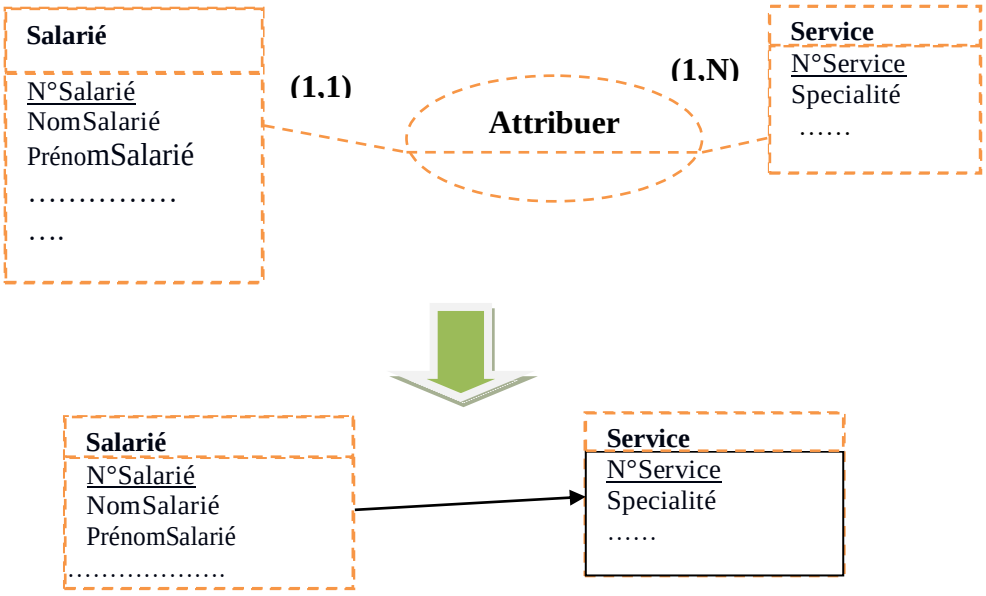


Figure 3.3 : Association Père-Fils

L'entité père ici c'est l'entité Service car elle a la cardinalité (1,N), ce type

d'association se traduit donc de l'immigration de la clé primaire de l'entité fils (Salarié) vers l'entité père (Service) , On aura donc comme résultat le modèle suivant :

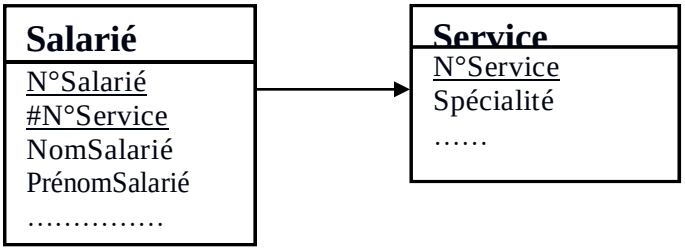


Figure 3.4 : Résultat obtenu du modèle

Remarque : #N°service de la table Salarié et qui fait référence de la table service, le signe « # » désigne la représentation de la clé étrangère. Ci-dessous un exemple pratique :

Service	
N°Service	Spécialité
1	Vente
2	Stock
3	Achat
…..	…..

Salarié			
N° Salarié	N°Service	Nom Salarié	Prénom Salarié
1	1	Amrane	Amira
2	1	Bendjamaa	Omar
3	2	KhaledKhodja	Adel
…..	…..	……	…..

Par exemple, nous avons le salarié Bendjamaa Omar qui a N°Service 1 appartient au service Vente

- **Règle 3** : « Association de type N:N » Une association binaire de type N : N devient une table supplémentaire dont la clé primaire est composée de la concaténation des deux clés étrangères (qui référencent les deux clés primaires des deux tables en associations) les propriétés de l'association deviennent des colonnes (Attributs) de cette nouvelle table. Voici un exemple d'illustration :

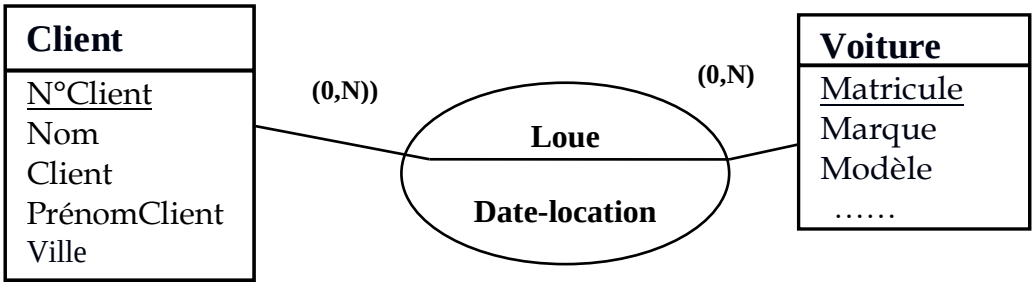


Figure 3.5 : Association N:N

Ici nous avons une association porteuse (Date location), on ne peut pas déterminer la date de location si on ne connait pas N°Client et Matricule voiture.

En appliquant la règle 3 on aura donc le résultat suivant :

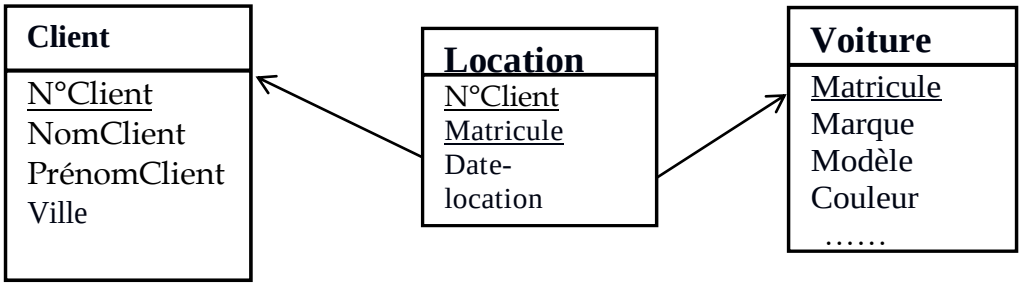


Figure 3.6 : Résultat obtenu de la règle 3

Voici un extrait de la table Client, Voiture et Location.

CLIENT			
N°Client	Nom Client	Prénom Client	Ville
1	Alaoui	Mounib	Annaba
2	Layeb	Souha	Setif
3	Mimoune	Oussama	Alger
…..	…..	……	…..

Location		
N°Client	Matricule	Date-location
1	2321A00	22/02/2020
3	1945B08	15/08/2021
1	2321A00	21/01/2020
…..	…..	……

Voiture			
Matricule	Marque	Modèle	Couleur
2321A00	Nissam	Micra	Noir
1945B08	Toyota	Yaris	Gris
1692C91	Renaud	Clio3	blanche
…..	…..	……	…..

A travers cette modélisation on peut savoir d'après la table Location les clients ayant loué une voiture !

- **Règle 4 (**Association **n-aires)** : L'association devienne une table et la clé de cette table est la concaténation des identifiants des entités reliées par l'association.

Elle a presque le même principe de la règle 3.

Ci-dessous un exemple d'illustration :

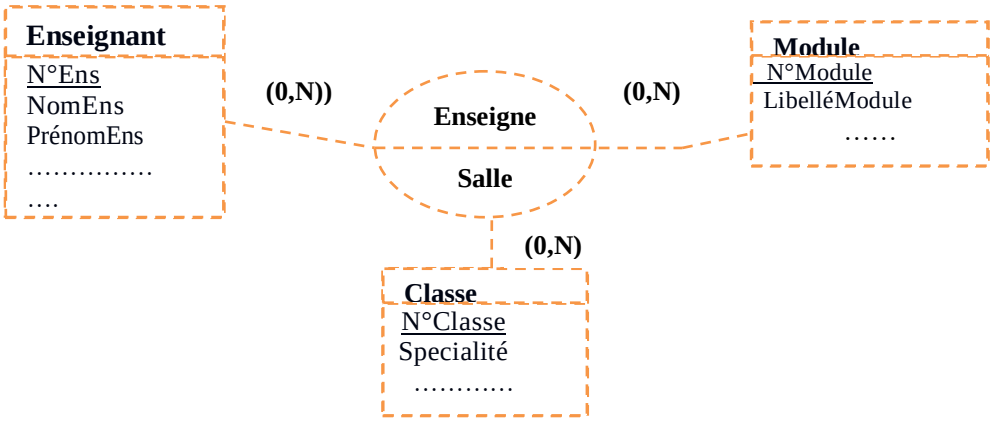


Figure 3.7 : Association n-aires

La transformation vers le modèle MLD s'effectue de la manière suivante : Comme illustré la figure ci-dessous, on aura une nouvelle table enseignement avec clé primaire égale à la concaténation des autres tables (N°Ens, N°Module, N°Classe). On garde la propriété comme attribut (salle).

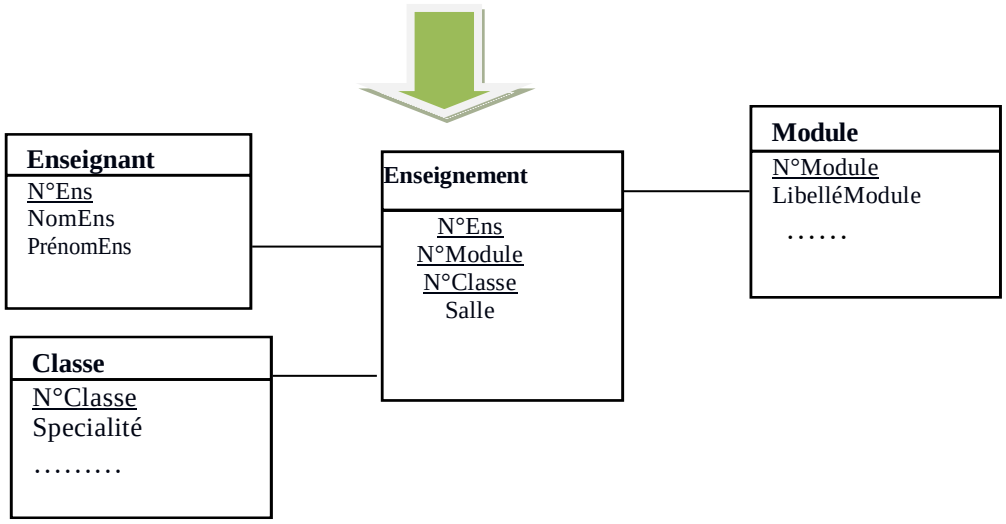


Figure 3.8 : Le modèle MLD obtenu après l'application de la Règle 4

Voici un exemple d'application pour mieux comprendre le déroulement de la règle 4 :

Module	
N°Matière	Libellé
1	Informatique
2	Mathématique
3	Physique
…..	…..

Classe	
N°Classe	Spécialité
1	Math & Informatique
2	Eléctronique
3	Génie Civil
…..	…..

Enseignant		
N°Ens	Nom	Prénom
1	Bey	Ali
2	Merdaci	Moussa
3	Merzougui	Karim
…..	…..	

Enseignement			
N°Ens	N°Matière	N°Classe	Salle
1	1	1	9
2	1	2	10
1	1	3	11
…..			…..

D'après la table un enseignant ayant le N°1 qui est Bey Ali enseigne le module informatique « N°1 » pour la classe Math & Informatique dans la salle 9.

Règle 5:

La solution la plus simple et la plus générale pour transférer une association 1 :1 consiste à traiter cette association 1 :1 comme une association 1 : N. Dans ce cas l'association disparait, la clé primaire de fils 1 devienne clé étrangère de fils 2 ou la clé primaire de fils 2 devienne clé étrangère de fils 1.

Soit l'exemple suivant :

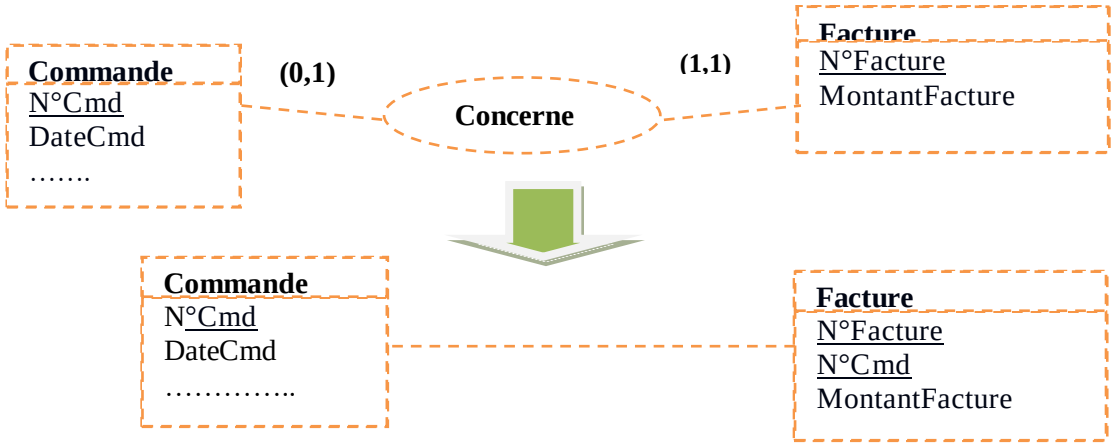


Figure 3.9 : Exemple de la règle 5

Pour clarifier la règle 5 voici un exemple d'application :

Commande	
N°Cmd	DateCommande
1	5/12/2020
2	8/1/2020
3	11/5/2021
…..	…..

Facture		
N°Facture	N°Cmd	MonatantFacture
001	1	2100000,00
002	2	13500,00
003	3	2500,00
…..	…..	……

Chaque commande est associée à une Facture.

Chapitre 4 : Algèbre Relationnelle

1. Définition de l'algèbre relationnelle

L'algèbre relationnelle a été inventée par E.Codd en 1970 dont le but de formaliser les opérations sur les ensembles. Elle constitue une collection d'opérations formelles qui agissent sur des relations et produisent des relations ;
Ces opérations sont regroupées, selon leurs caractéristiques, en plusieurs familles.
L'algèbre relationnelle est à la base du langage d'interrogation SQL implémenté sur les SGBD actuels.

2. Les opérations ensemblistes

Les opérations ensemblistes sont appliquées à deux relations et permettent de générer une troisième relation. Elles sont donc binaires et déduites de théories des ensembles.

2.1.Union

L'union est une opération sur deux relations de même schéma R1 et R2 qui sert à construire une troisième relation R3 de même schéma ayant comme tuples ceux appartenant à R1, à R2 ou aux deux.
Les tuples qui apparaissent plusieurs fois dans le résultat ne sont représentés qu'une seule fois (pas de doublons)

Notation

- R= R1 ∪ R2
- ***UNION (R1, R2)***
- ***APPEND*** *(RelationR1, RelationR2)*

Représentation graphique

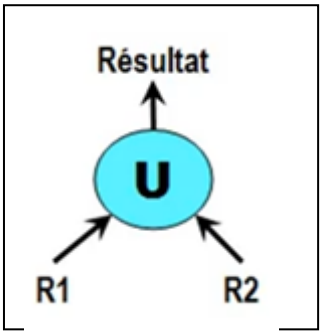


Figure 4.1 : Représentation graphique de l'union

Exemple : Soit les deux relations R1 et R2 dont l'extension est comme suit.

R1

N°	Nom	TEL

1	Amir	021152535

R2

N°	Nom	TEL
2	Akram	021182639

R= R1 ⋃ R2

N°	Nom	TEL
1	Amir	021152535
2	Akram	021182639

2.2. Produit cartésien

Le produit cartésien de deux relations R1 et R2 de schéma quelconque est une relation R3 ayant pour attributs la concaténation des attributs de R1 et de R2 et dont les tuples sont constitués de toutes les concaténations d'un tuple de R1 à un tuple de R2

Notations

- R1 **X** R2
- **PRODUCT (R1, R2)**
- **TIMES (R1, R2)**

Représentation graphique

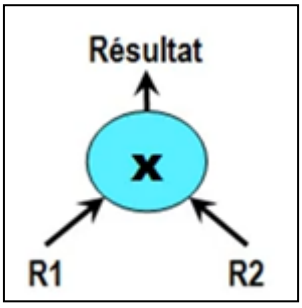


Figure 4.2 : Représentation graphique de produit cartésien

R1

N°	Nom	TEL
1	Amir	021152535
2	Akram	021182639

R2

Cmd	Date
4	04/04/2021

R3= R1 X R2

N°	Nom	Cmd	TEL	Date
1	Amir	4	021152535	04/04/2021
2	Akram	4	021182639	04/04/2021

Note
Dans le cas ou les deux opérations ont des attributs ayant les même noms, on représente au niveau du résultat ces attributs avec d'autres nom ou bien en spécifiant la relation à laquelle ils appartiennent : R1.A, R2.A.

3. Les opérations spécifiques

En plus des opérations ensemblistes, un ensemble d'opérations dites spécifiques sont définies dans l'algèbre relationnelle. Certaines sont binaires et d'autres unaires et permettent également d'effectuer des transformations entre les données des relations afin d'en déduire de nouvelles relations répondant à nos besoins en information

3.1. Projection

La projection d'une relation R(A1,A2,…,An) sur les attributs Ai, Ai+1,…,Ap (avec p<n) est une relation R2 de schéma Ai, Ai+1,…,Ap et dont les tuples sont obtenus par élimination des attributs de R n'appartenant pas à R2 et par suppression des doublons.

Notations

- Π A1, A2… Am (R1)
- R1 [Ai, Aj... Am]
- PROJECT (R1, A1, A2,…, Am)

Notation graphique

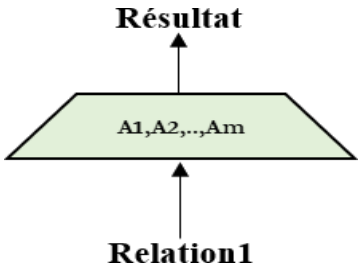


Figure 4.3 : Représentation graphique de la projection

Exemple :
Soit la relation Employé (CodeE, NomE, Tel-E) En extension :

CodeE	NomE	Tel-E
E101	Mounir	021322100
E102	Foued	021554431

NomE
Mounir
Foued

ΠNomE (Employé)

3.2. Restriction (Sélection)

La restriction (ou sélection) de la relation R par une condition C est une relation R2 de même schéma dont les tuples sont ceux de R satisfaisant la condition C.
La condition est de la forme <Attribut> Opérateur<Valeur>
Les opérateurs sont {=,<,>,<=,>=, <>}

Notations

- σ Condition (R)
- R1 [Condition]
- RESTRICT (R, Condition)

Représentation graphique

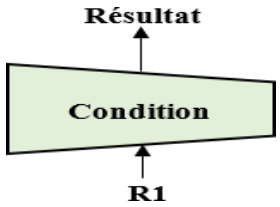


Figure 4.4 : Représentation graphique de la sélection

Exemple :

Soit la table Etudiant en extension :

CodeE	NomE	AgeE
E101	Mounir	35
E102	Foued	21
E103	Leila	28
E104	Meryem	41

σAgeE > 30 (Etudiant)

CodeE	NomE	AgeE
E101	Mounir	38
E104	Meryem	40

3.3. Thêta Jointure

La thêta-jointure de deux relations R1 et R2 de schéma quelconque selon une condition C est une relation R3 dont le schéma est la concaténation des attributs des deux relations et les tuples sont ceux du produit cartésien entre R1 et R2 respectant la condition C.
La condition C est de la forme <Attribut> opérateur <Attribut>.
Les opérateurs peuvent être arithmétiques (=, >, <, <=, >=, <>) ou logique (Et, Ou, Non).

Notation

JOIN (R1, R2, Condition)

Représentation graphique

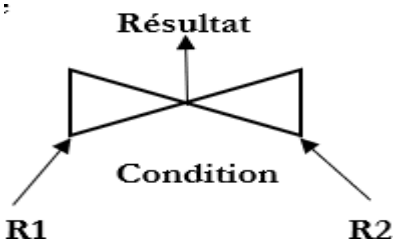


Figure 4.5 : Représentation graphique de la jointure

Exemple :

Employé

Nom E	Salaire E
Taleb	20000
Mansouri	10000
Bouraoui	6000

Chef

Num C	Salaire C
Benmohamed	25 000
Bendjama	12 000

Employé ⋈ **Chef**

Salaire E > Salaire C

Nom E	Salaire E	Nom C	Salaire C
Taleb	**20000**	Benmohamed	**12000**

Note : si l'opérateur est = alors c'est une Equi-jointure sinon c'est une Inéqui-jointure.

3.4. Jointure Naturelle

La jointure naturelle de deux relations R1 et R2 de schéma quelconque donne une troisième relation R3 dont le schéma est obtenu avec concaténation des attributs de R1 et ceux de R2 mais en ne prenants les attributs de même nom qu'une seule fois. Les tuples de R3 sont ceux de R1 et R2 respectant une equi-jointure entre les attributs de même nom.

Notation
JOIN (R1, R2)

Représentation graphique

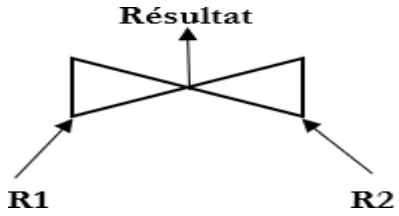


Figure 4.6 : Représentation graphique de la jointure naturelle

Exemple :

Ami

Nom	Prénom	Age
Taleb	Amira	6
Mansouri	Ali	40
Bouraoui	Farid	20
Mimoun	Mouna	6

Cadeau

Age	Article	Prix
40	Parfun	120
6	Montre	320
20	Garniture	82

R=⋈
Ami Cadeau

Nom	Prénom	Age	Article	Prix
Taleb	Amira	6	Montre	320
Mansouri	Ali	40	Parfun	120
Bouraoui	Farid	20	Garniture	82
Mimoun	Mouna	6	Montre	320

Note : Une jointure naturelle entre deux relations R1 et R2 n'ayant aucun attribut en commun (de même nom) est le produit cartésien de R1 et de R2.

4. Les opérations dérivés

A partir des opérations vues précédemment, on peut en déduire d'autres opérations afin de les utiliser pour simplifier les opérations sur les données. Ces opérations sont dites dérivées.

4.1 Intersection

Définition

L'intersection de deux relation R1 et R2 de même schéma est une relation R3 de même schéma dont les tuples sont ceux appartenant à la fois à R1 et à R2

Notation

- R1 ∩ R2
- INTERSECT (R1, R2)
- AND (R1, R2)

Représentation graphique

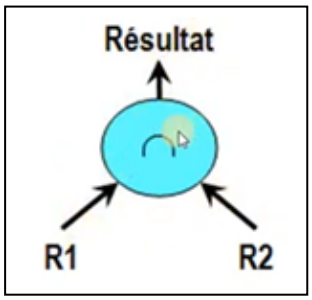


Figure 4.7 : Représentation graphique de l'intersection

Exemple :

Considérant l'exemple suivant :

Deux bibliothèques B1 et B2 fusionnent et décident de rechercher les livres qu'elles ont en communpour n'en garder un seul exemplaire. Chaque bibliothèque possède une relation LivreB1 et LivreB2.

LivreB1

NLIV	TITRE	NOMAUT
501	Mort sur le Nil	Agatha Christie
502	Ils étaient dix	Agatha Christie
610	Le Crime de l'Orient-Express	Agatha Christie
301	Les 10 mousquetaires	Artagnan

LivreB2

NLIV	TITRE	NOMAUT
101	La basilique de Paris	Victorien
502	Ils étaient dix	Agatha Christie
210	Le roi de la forêt	Aiglon
301	Les 10 mousquetaires	Artagnan

LivreB2 ∩ LivreB2= INTERLIVRE

NLIV	TITRE	NOMAUT
502	Ils étaient dix	Agatha Christie
301	Les 10 mousquetaires	Artagnan

INTERLIVRE contient donc toutes les tuples communes aux deux tables **LivreB1**et **LivreB2**

4.2. Division

La division de la relation R1(A1, A2, …, An) par la sous-relation R2(Ap+1, …, An) est la relation R3(A1, A2, …, Ap) formée de tous les tuples qui concaténés à chaque tuple de R2 donnent toujours un tuples de R1.

Notation
R1/R2

DIVISION (R1, R2)

Représentation graphique :

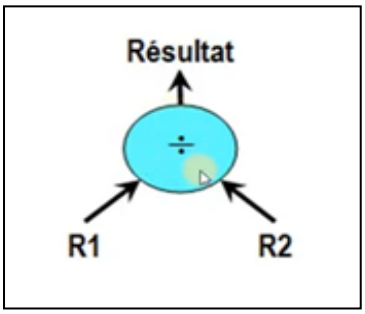


Figure 4.8 : Modélisation graphique de la division

R1

Nom	Age	Ville
Ali	12	Alger
Ali	23	Oran

Ahmed	**12**	**Alger**
Ahmed	**23**	**Alger**
Malik	**43**	**Alger**

R2

Age	**Ville**
12	**Alger**
23	**Alger**

R3= R1 / R2

Nom
Ahmed

Note : les attributs du résultat d'une division sont ceux faisant partie de la première relation et ne sont pas dans la seconde pour que le produit cartésien du résultat avec la deuxième donnent tous les attributs de la première relation.
Pour effectuer une division entre R1 et R2 il faut que tous les attributs de R2 font partie de R1 et que R1 possède au moins un attribut en plus que R2.

4.3. Jointure externe

La jointure externe entre deux relations R1 et R2 de schéma quelconque est une relation R3 dont le schéma est la concaténation des attributs de R1 et de ceux de R2 en ne représentant les attributs ayant le même nom qu'une seule fois. Les tuples de R3 sont ceux obtenus avec une jointure naturelles entre R1 et R2 et ceux de R1 et de R2 ne participants pas à la jointure en représentant par des valeurs nulles ceux de l'autre relation.

Notation
EXT-JOIN (R1, R2)
Représentation graphique

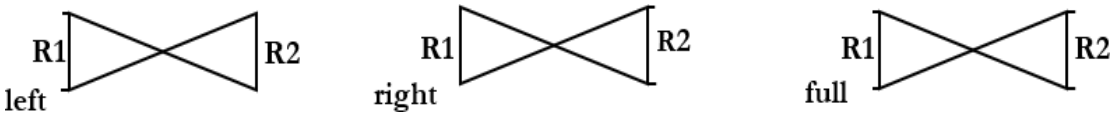


Figure 4.9 : Représentation graphique de la jointure externe

Note : On distingue deux autres variantes de la jointure externe, la jointure externe droite et la jointure externe gauche notées respectivement REXT-JOIN et LEXT-JOIN. La première donne tous les attributs de la relation à droite de la jointure externe et uniquement ceux de la relation gauche qui participent à la jointure. La seconde c'est l'inverse.

Exemple **:**

R1

NA	Nom	NumTel
1	Mohamed	032482223
2	Ahmed	032485760
3	Mohammed	032488080

R2

NB	**NumClient**	**Adresse**
0	Mohamed	Alger
2	Malik	Blida
5	Malik	Oran

R3= **EXT-JOIN (R1, R2)**

NA	Nom	Tel	NB	Adresse
1	Mohamed	032482223	0	Alger
2	Ahmed	032485760	-	-
3	Mohamed	032488080	0	Alger
-	Malik	-	2	Blida
-	Malik	-	5	Oran

4.4. Semi-jointure

La semi-jointure de deux relations R1 et R2 de schéma quelconque est une relation R3 dont le schéma est celui de R1 et les tuples sont ceux de R1 appartenant à la jointure naturelle entre R1 et R2 ;

Notation

R1 ⋉ **R2**
SEMI-JOIN (R1, R2)

Représentation graphique

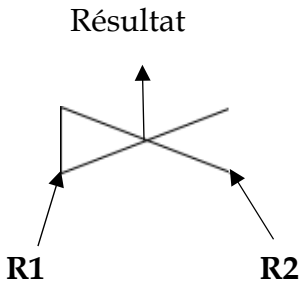


Figure 4.10 : Modélisation graphique de Semi jointure

Exemple :

R1

NA	Nom	NumTel
1	Mohamed	032482223
2	Ahmed	032485760
3	Mohammed	032488080

R2

NB	NumClient	Adresse
0	Mohamed	Alger
2	Malik	Blida
5	Malik	Oran

R1 R2

NA	Nom	NumTel
1	Mohamed	032482223
3	Mohamed	032488080

Chapitre 5 : Le Langage SQL

1. Introduction

Le langage SQL (Structured Query Langage) peut être considéré comme le langage d'accès normalisé aux bases de données. Il est supporté par la plupart des produits commerciaux que ce soit par les systèmes de gestion de bases de données micro tel qu'access ou par les produits plus professionnels tels qu'Oracle.

Le succès du langage SQL est dû essentiellement à sa simplicité et au fait qu'il s'appuie sur le schéma relationnel pour énoncer des requêtes en laissant le SGBD responsable de la stratégie d'exécution. De manière synthétique, on peut dire que SQL est un langage permettant la manipulation et la communication avec une base de données.

L'intérêt de ce langage est qu'il est très bien documenté, il permet, via des requêtes, de créer, interroger, modifier, supprimer et manipuler les données.

2. Fonctionnalités

SQL est un langage composé de sous ensembles :

✓ **LDD** : Le langage de définition de données (DDL : Data Definition Language) pour manipuler les structures de données d'une base de données. Exemple de commandes : **CREATE, DROP, ALTER**.

CREATE : Créer d'une structure de données ;

DROP Suppression d'une structure de données et **ALTER** pour la modifier .

✓ **LMD** : Le langage de Manipulation de données (DML : Data Manipulation Language) pour la recherche, l'insertion, la mise à jour et la suppression de données. Le LMD est basé sur les opérateurs relationnels, auxquels sont ajoutés des fonctions de calcul d'agrégats et des instructions pour réaliser les opérations d'insertion, mise à jour et suppression.

Exemple de commandes **: INSERT, UPDTE, DELETE, SELECT**

INSERT : Pour insérer des données dans une Base de Données Relationnelle, **UPDATE** pour les modifier, **DELETE** pour les supprimer et **SELECT** pour afficher les données

✓ **LCD** : le langage de contrôle de données (DCL : Data Control Language) pour contrôler l'accès aux données d'une base de données.

Exemple de commandes **: GRANT, REVOKE**

GRANT : Autorisation d'un utilisateur à effectuer une action ;

REVOKE : Annulation d'une commande de contrôle de données précédente.

3 Langage de Manipulation de Données (LMD)

3.1 Projection

La principale commande du langage de manipulation de données est la commande SELECT.

3.2 La commande SELECT

La commande SELECT permet de rechercher des données à partir de plusieurs tables; le résultat est présenté sous forme d'une table réponse.
Cette commande peut sélectionner un ou plusieurs attributs d'une table.
L'utilisation basique de cette commande s'effectue de la manière suivante :

```
SELECT "nom-attribut"
FROM "nom-table";
```

Cette requête va sélectionner (SELECT) l'attribut "nom- attribut" provenant (FROM) de la table appelée "nom-table".
Exemple : on considère l'extrait de la table « Client » :

Id	Nom	Prénom	Ville
1	Pierre	Dupond	Paris
2	Sabrina	Durond	Nantes
3	Julien	Martin	Lyon
4	David	Bernard	Marseille
5	Marie	Leroy	Grenoble

Si on veut avoir la liste de toutes les villes des clients, il suffit d'effectuer la requête suivante :

SELECT Ville
FROM client --→

Ville
Paris
Nantes
Lyon
Marseille
Grenoble

Avec la même table « client » il est possible de lire plusieurs attributs à la fois. Il suffit de séparer les noms des attributs souhaités par une virgule.

Pour obtenir les prénoms et les noms des clients :

SELECT nom, Prénom
FROM client -→

Nom	Prénom
Pierre	Dupond
Sabrina	Durond
Julien	Martin
David	Bernard
Marie	Leroy

SELECT *
FROM client

Id	Nom	Prénom	Ville
1	Pierre	Dupond	Paris
2	Sabrina	Durond	Nantes
3	Julien	Martin	Lyon
4	David	Bernard	Marseille
5	Marie	Leroy	Grenoble

-→

Il est possible de retourner automatiquement les attributs d'une table sans avoir à connaitre le nom de tous les attributs. Au lieu de lister toutes les attributs, il faut utiliser le caractère « * ».

3.3. La commande DISTINCT

L'utilisation de la commande SELECT en SQL permet de lire toutes les données, d'une ou plusieurs attributs. Cette commande peut potentiellement afficher des tuples en doubles. Pour éviter des redondances dans les résultats il faut ajouter DISTINCT après le mot SELECT.

Syntaxe :

SELECT DISTINCT « Nom-champ »
FROM « nom-table »

Exemple : Soit l'extrait de la table personne suivante :

Id	Nom	Prénom
1	Pierre	Dupond
2	Sabrina	Durond
3	Julien	Martin
4	David	Bernard
5	Julien	Martin

SELECT nom
FROM personne

Nom
Pierre
Sabrina
Julien
David
Julien

SELECT DISTINCT nom
FROM personne

Nom
Pierre
Sabrina
Julien
David

3.4. La Restriction

Une restriction consiste à sélectionner les tuples satisfaisant à une condition logique effectuée sur leurs attributs.

La commande **WHERE** dans une requête SQL permet d'extraire les tuples d'une base de données qui respectent une condition. Cela permet d'obtenir uniquement les informations désirées.

Syntaxe:

```
SELECT "nom-attribut"
FROM "nom-table"
WHERE "condition";
```

Exemple: on considère l'extrait de la table "client" suivante:

Id	Nom	Nbr-Commande	Ville
1	Pierre	4	Paris
2	Sabrina	0	Nantes
3	Julien	1	Lyon
4	David	2	Marseille
5	Julien	7	Paris

Pour obtenir toutes les infos sur les clients de Paris :

```
SELECT *
FROM client
WHERE Ville ='Paris';
```

Id	Nom	Nbr-Commande	Ville
1	Pierre	4	Paris
5	Julien	7	Paris

Il existe plusieurs opérateurs à utiliser avec la clause WHERE. La liste ci-jointe présente les opérateurs les plus couramment utilisés.
Les opérateurs logiques tel que :

- AND
- OR
- NOT

Les comparateurs logiques tel que :

- IN
- BETWEEN
- LIKE

Les opérateurs arithmétiques tel que : -, +, * , / , % , ….
Les comparateurs arithmétiques tel que : = , != , > , < , >= , <=, ….

3.5. Opérateurs logiques (AND, OR)

Une requête SQL peut être restreinte à l'aide de la condition **WHERE**. Les opérateurs logiques **AND** et **OR** peuvent être utilisées au sein de la commande WHERE pour combiner des conditions.

L'opérateur **AND** permet d'assurer que la condition1 ET la condition2 sont vraies :

```
SELECT nom_attribut
FROM nom_Table
WHERE Condition1 AND Condition2 ;
```

L'opérateur **OR** permet d'assurer que la condition1 OU la condition2 sont vraies :

```
SELECT nom_attribut
FROM nom_Table
WHERE Condition1 OR Condition2 ;
```

Ces opérateurs peuvent être combinés à l'infini :

```
SELECT nom_attribut
FROM nom_Table
WHERE Condition1 AND (Condition2 OR Condition3);
```

Pour illustrer les prochaines commandes, nous allons considérer l'extrait de la table « Produit » suivante :

id	Nom	Catégorie	Stock	Prix
1	Ordinateur	Informatique	8	150
2	Clavier	Informatique	40	40
3	Souris	Informatique	18	35
4	Gomme	Fourniture	187	5

```
SELECT *
FROM Produit
WHERE Catégorie = 'Informatique' AND stock<20;
```

Selon cette requête, on cherche à afficher la liste des produits mais selon deux critères qui sont Catégorie = 'Informatique' et stock<20. Le résultat de cette requête est la suivante :

id	Nom	Catégorie	Stock	Prix
1	Ordinateur	Informatique	8	150

Cette fois ci, on s'intéresse à afficher les produits dont la condition est Catégorie = 'Ordinateur' **ou** nom = 'clavier'. La requête en question est la suivante :

```
SELECT *
FROM Produit
WHERE Catégorie = 'Ordinateur' OR nom ='clavier' ;
```

Voici le résultat après exécution de cette requête :

id	**Nom**	**Catégorie**	**Stock**	**Prix**
1	Ordinateur	Informatique	8	150
2	Clavier	Informatique	40	40

La syntaxe d'une requête qui combine les deux opérateurs OR et AND est la suivante :

```
SELECT *
FROM Produit
WHERE (Categorie = 'Informatique' AND prix >=40) OR
(categorie = 'fourniture' AND stock<200) ;
```

Nous donnons ci-dessous un exemple combiné les deux opérateurs OR et AND.

id	Nom	C	Stock	Prix
1	Ordinateur	Informatique	8	150
2	Clavier	Informatique	40	40
4	Gomme	Fourniture	187	5

3.6. Opérateur IN

L'opérateur logique IN dans SQL s'utilise avec la commande WHERE pour vérifier si un attribut est égal à une des valeurs comprise dans un ensemble déterminé de valeurs.

La requête équivalente avec l'opérateur **IN** est la suivante :

```
SELECT nom- attribut
FROM nom-table
WHERE nom- attribut IN (valeur1, valeur2, valeur3, …) ;
```

Exemple : On considère l'extrait de la table « adresses » qui contient une liste d'adresse associée à des utilisateurs d'une application.

Id	Prénom utilisateur	Adresse	Code postal	Ville
1	Mohamed	Philippe de cerner, Menadia	23000	Annaba
3	Ali	Rue Kanouni Athmane	69000	Canstantine
4	Lina	Rue Didouche Mourad	13000	Alger

Si on souhaite obtenir la liste des utilisateurs de la ville d'Annaba et d'Alger :

SELECT *
FROM Adresse
WHERE Ville **IN** ('Annaba', 'Alger ');
On aura après exécution de la requête le résultat suivante.

Id	Id-utilisateur	Adresse	Code postal	Ville
1	Mohamed	Philippe de cerner, Menadia	23000	Annaba
4	Lina	Rue Didouche Mourad	16000	Alger

3.7. Opérateur BETWEEN

L'opérateur BETWEEN est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant WHERE. L'intervalle peut être constitué de chaines de caractères, de nombres ou de dates.
Syntaxe l'opérateur BETWEEN est le suivant :

```
SELECT *
FROM table
WHERE nom-attribut BETWEEN « valeur1 » AND « valeur2 »;
```

La requête suivante retournera toutes les tuples dont la valeur de l'attribut « nom-attribut » comprise entre valeur1 et valeur2.

Exemple 1 : Considérons la table « utilisateur » qui contient les membres d'une application en ligne :

id	Nom	Date inscription
1	Pierre	25-01-2021
2	Sabrina	11-03-2021
3	Julien	22-03-2021
4	David	14-02-2021
5	Marie	09-04-2021

Si on souhaite obtenir les membres qui se sont inscrit entre le 05 mars 2021 et le 25 mars 2021

SELECT *
FROM utilisateur
WHERE Date inscription **BETWEEN** '05-03-2021' **AND** '25-03-2021';

On aura le résultat suivant :

id	Nom	Date inscription
2	Sabrina	11-03-2021
3	Julien	22-03-2021

Exemple 2 :
Si on souhaite obtenir tous les résultats dont l'identifiant n'est pas situé entre 4 et 10.

```
SELECT *
FROM  utilisateur
WHERE  id NOT BETWEEN 4 AND 10 ;
```

id	Nom	Date inscription
1	Pierre	25-01-2021
2	Sabrina	11-03-2021
3	Julien	22-03-2021

3.8. Opérateur LIKE

L'opérateur LIKE est utilisé dans la clause WHERE des requêtes SQL. Ce mot-clé permet d'effectuer une recherche dans la base de données.
Syntaxe:

```
SELECT *
FROM table
WHERE attribut LIKE modèle ;
```

Un modèle permet de définir un critère de filtrage des données selon sa structure.
Un modèle ressemble généralement à l'un des exemples suivants :

- **LIKE '%a'** : Le caractère « % » est un caractère joker qui remplace tous les autres caractères. Ainsi, ce modèle permet de rechercher toutes les chaines de caractère qui se termine par un « a ».
- **LIKE 'a%'** : Ce modèle permet de rechercher toutes les lignes de « colonne » qui commence par un « a ».
- **LIKE '%a%'** : Ce modèle est utilisé pour rechercher tous les enregistrements qui utilisent le caractère «a ».
- **LIKE 'pa% on'** : Ce modèle permet de rechercher les chaines qui commences par « pa » et se terminent par « on », comme « pantalon » ou « pardon ».
- **LIKE 'a_c'** : Le caractère « _ » peut être remplacé par n'importe quel

caractère mais un seul caractère uniquement. Ainsi ce modèle permet de retourner les tuples « aac », « abc » ou même « azc ».

Exemple : On considère la table « client » ci-dessous :

id	Nom	Ville
1	Alice	Dijon
2	Bob	Lille
3	Daniel	Paris
4	Francois	Metz
5	Marie	Grenoble

Les clients habitant une ville qui commence par « A » sont sélectionnés dans le tableau résultat :

Id	Nom	Ville
1	Alice	Dijon
4	Bob	Lille
5	Marie	Grenoble

La requête est la suivante :

```
SELECT *
FROM Client
WHERE Ville LIKE 'A%';
```

Les clients habitant une ville qui termine « r ».

id	Nom	Ville
1	Alice	Dijon
4	François	Metz

```
SELECT *
FROM Client
WHERE Ville LIKE '% r';
```

3.9. Les Opérateurs IS NULL/IS NOT NULL

Dans le langage SQL, l'opérateur IS permet de filtrer les résultats qui contiennent la valeur NULL. Cet opérateur est indispensable car la valeur NULL est une valeur inconnue et ne peut par conséquent pas être filtrée par les opérateurs de comparaison (égale, inférieure, supérieure ou différente).

Syntaxe:

```
SELECT *
FROM « table »
WHERE nom-attribut IS NULL ;
```

A l'inverse, pour filtrer les résultats et obtenir uniquement les valeurs qui ne sont pas nuls.

```
SELECT *
FROM 'Table'
WHERE nom-attribut IS NOT NULL ;
```

Exemple:
On considère une application qui possède un extrait d'une table contenant les utilisateurs. Cette table possède 2 attributs pour associer les adresses de livraison et de facturation ou de livraison alors le champ reste à NULL.

Id	Nom	Date inscription	@ Livraison_id	@ Facturation_id
1	Pierre	25-01-2021	12	12
2	Sabrina	11-03-2021	Null	Null
3	Julien	22-03-2021	13	14
4	David	14-02-2021	Null	Null

Utilisateur sans adresse de livraison :

```
SELECT *
FROM utilisateur
WHERE @ Livraison_id IS NULL ;
```

Utilisateur avec adresse de livraison :

```
SELECT *
FROM utilisateur
WHERE @ Livraison_id IS NOT NULL ;
```

3.10. Le tri des résultats

La commande ORDER BY permet de trier les tuples d'une requete SQL. Il est possible de trier les données sur une ou plusieurs attributs, par ordre ascendant ou descendant.

La syntaxe est le suivant:

```
SELECT attribut1, attribut2
FROM table
ORDER BY attribut1;
```

Par défaut les résultants sont classes par ordre ascendant, toutefois il est possible d'inverser l'ordre en utilisant le suffixe DESC après le nom de l'attribut. Par ailleurs, il est possible de trier sur plusieurs attributs en les séparant par une virgule.

```
SELECT attribut1, attribut2, attribut3,
FROM table
ORDER BY attribut1 DESC, attribut2 ASC ;
```

Exemple : Soit l'extrait de la table utilisateur suivante :

id	Nom	Prenom	Date inscription	Tarif
1	Pierre	Dupond	25-01-2021	125
2	Sabrina	Durand	11-03-2021	42
3	Julien	Martin	22-03-2021	55
4	David	Bernard	14-02-2021	27
5	Marie	Leroy	19-06-2020	98

Pour récupérer la liste des utilisateurs classés par ordre alphabétique du Nom on utilise :

```
SELECT *
FROM utilisateur
ORDER BY Nom ;
```

Après exécution de cette requête on aura le résultat suivant :

id	Nom	Prénom	Date inscription	Tarif
5	Marie	Leroy	19-06-2020	98
1	Pierre	Dupond	25-01-2021	125
4	David	Bernard	14-02-2021	27
2	Sabrina	Durand	11-03-2021	42
3	Julien	Martin	22-03-2021	55

Exemple 2 : Si on souhaite afficher la liste des utilisateurs classée par ordre alphabétique du nom et pour ceux qui ont le même nom de famille, les trier par

ordre décroissant d'inscription.

La syntaxe de la commande **ORDER BY est la suivante** :

```
SELECT *
FROM utilisateur
ORDER BY Date_inscription DESC ;
```

Voici le résultat :

id	Nom	Prenom	Date inscription	Tarif
3	Julien	Martin	22-03-2021	55
2	Sabrina	Durand	11-03-2021	42
4	David	Bernard	14-02-2021	27
1	Pierre	Dupond	25-01-2021	125
5	Marie	Leroy	19-06-2020	98

3.11. Regroupement des résultats

3.11.1. La commande GROUP BY

Il peut être intéressant de regrouper des résultats afin de faire des opérations par groupe (opérations statistiques par exemple). Cette opération se réalise à l'aide de la clause **GROUP BY**, suivie du nom de chaque attribut sur laquelle on veut effectuer des regroupements.

La syntaxe de la commande GROUP BY est la suivante :

```
SELECT attribut1, fonction (attribut2)
FROM table
GROUB BY attribut1 ;
```

Voici quelques fonctions :

- **AVG**: Calcule la moyenne d'un set de valeurs
- **COUNT**: Calcule le nombre de tuples
- **MAX**: Récupérer la valeur maximale
- **MIN**: Récupérer la valeur minimale
- **SUM**: Calcule la somme d'un set de valeurs

Exemple:
Prenons en considération l'extrait de la table "achat" qui résume les ventes d'une boutique.

ID	Client	Tarif	Date-achat
1	Mohamed	52	11-08-2020
2	Mustapha	55	15-08-2020
3	Ali	82	31-04-2020
4	Mustapha	33	15-02-2020
5	Mohamed	17	05-04-2020

Pour obtenir le coût total de chaque client :
SELECT client, SUM (tarif)
FROM achat
GROUP BY client ;

Client	SUM (tarif)
Mohamed	69
Mustapha	88
Ali	82

3.11.2. La commande HAVING

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM (), COUNT(), AVG(), MIN(), MAX().

La syntaxe de la commande HAVING **est la suivante :**

SELECT *
FROM nom-table
GROUB BY attribut1
HAVING function (attribut 2) **operateur** valeur ;

Cela permet donc de sélectionner toutes les attributs de la table 'nom-table' en groupant les tuples qui ont des valeurs identiques sur l'attribut " attribut 1" et que la condition de **HAVING** soit respectée.

Exemple :
Soit l'extrait de la table « achat » suivante :

ID	NomClient	Tarif	Date-achat
1	Pierre	102	11-08-2020
2	Marie	55	15-08-2020
3	Sabrina	18	31-04-2020
4	David	20	15-02-2020
5	Pierre	160	05-04-2020

Si on souhaite récupérer la liste des clients qui ont commandé plus de 50 euros. On utilise la requête suivante :

```
SELECT client, SUM (tarif)
FROM achat
GROUP BY client
HAVING SUM (tarif) > 40 ;
```

Voici le résultat de la table:

ID	NomClient	SUM (Tarif)
1	Pierre	262
2	Marie	55

3.12. La jointure

Les jointures en SQL permettent d'associer plusieurs tables dans une même requête. Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.

Une jointure est un produit cartésien de deux tables. On appelle équijointure une jointure dont la qualification est une égalité entre deux attributs. En SQL, l'expression d'une jointure se fait en précisant le nom des tables sur lesquelles on fait la jointure. La clause WHERE permet de préciser la quantification de la jointure.

Exemple :

Voiture

Marque	**Modèle**	**Série**	**Compteur**	**Num propriétaire**
Peugeot	108	KID	8514HV57	123450
Peugeot	301	KID	6576VE38	560000
Peugeot	508	Chorus	7845ZS83	12000
Fiat	Panda 4x4	GT1	8941UD61	13000
Ford	Festa	Match	8562EV23	28700
Renault	Megane 5	RL	124560	75600
Renault	Clio3	RL	56000	75600
Audit	A4	Quatro	7846AZS75	21350

Société

Marque	**Pays**
Peugeot	France
Fiat	Italie
Ford	Etats unis
Renault	France

Audit	Turley

On veut afficher le pays d'origine des voitures par marque/modèle.
L'affichage des pays d'origine des voitures par marque/modèle se fait par la requête suivante :

```
SELECT Marque, Modèle, Pays
FROM Voiture, Société WHERE
Voiture. Marque = Société. Marque ;
```

Une autre version

```
SELECT  Voiture. Marque, Voiture. Modèle,  Société. Pays
FROM Voiture, Société
WHERE Voiture. Marque = Société. Marque ;
```

Marque	Modèle	Pays
Peugeot	108	France
Peugeot	301	Fnance
Peugeot	508	France
Fiat	Panda 4x4	France
Ford	Festa	Etats unis
Renault	Megane 5	France
Renault	Clio3	France
Audit	A4	Turkey

Il est aussi possible de donner des alias aux noms des tables pour diminuer la taille des requêtes. Par exemple :

```
SELECT V.Marque , V .Modèle, S.Pays
FROM Voiture V, Société  S
WHERE
V. Marque = S. Marque ;
```

3.13. Les sous requêtes

Effectuer une sous requête consiste à effectuer une requête à l'intérieur d'une autre, ou en d'autres termes d'utiliser une requête afin d'en réaliser une autre.
La syntaxe est la suivante:

```
SELECT …
FROM ….
WHERE Opérateur (SELECT ….. FROM ….);
```

Exemple : Soit les deux extraits des tables suivantes Etudiant et Filière:

Etudiant :

N°_Etudiant	Nom_Etud	Prenom_Etud	Date_Naiss	ID_Filière
E1	Julien	Martin	22-03-2021	**5**
E2	David	Bernard	11-03-2021	**5**
E3	Marie	Leroy	14-02-2021	**2**
E4	Pierre	Dupond	25-01-2021	**1**

Filière :

Nom_Filière	ID_Filière
Informatique	**1**
Droit	**2**
commerce	**3**
Anglais	**4**
Physique	**5**

La liste des filières n'ayant aucun étudiant.

```
SELECT *
FROM Filière
WHERE
IDFilière NOT IN
(SELECT Distinct IdFiliere
FROM Etudiant ) ;
```

Nom_Filière	ID_Filière
Droit	**2**
Physique	**5**

3.14. Les opérations ensemblistes

Les opérations ensemblistes en SQL, sont celles définies dans l'algèbre relationnelle (union, intersection, différence).

3.14.1. La commande UNION

La commande UNION de SQL permet de mettre bout à bout les résultats de plusieurs requêtes utilisant elles mêmes la commande SELECT. C'est donc une commande qui permet de concaténer les résultats de deux requêtes ou plus. Pour l'utiliser il est nécessaire que chacune des requêtes à concaténer retournes le même nombre de colonnes avec les mêmes types de données et dans le même ordre.

La syntaxe pour la commande UNION est la suivante:

```
SELECT *
FROM table1
UNION
SELECT *
FROM table2 ;
```

Par défaut, les données exactement identiques ne seront pas répétées dans les résultants. Pour effectuer une union dans laquelle même les tuples dupliquées sont affichées il faut utiliser la commande UNION ALL.

Exemple : Imaginons une entreprise qui possède deux magasins et dans chacun de ces magasins il y a une table qui liste les clients.

Magasin1-Client1

Nom	Prénom	Date-Naiss	Ville	Total
Bernard	David	27-10-1984	Metz	284
Durand	Sabrina	03-06-1996	Marseille	132
Martin	Julien	24-11-1977	Dijon	74
Daniel	François	11-04-1980	Paris	37

Magasin2-Client2

Nom	Prénom	Date-Naiss	Ville	Total
Bernard	David	27-10-1984	Metz	284
Pierre	Dupond	03-06-1990	Toulouse	130
Bob	Martin	10-01-1972	Lille	70
Daniel	François	11-04-1980	Paris	37

Sachant que certains clients sont présents dans les deux tables. Pour éviter de retourner plusieurs fois les mêmes tuples, il convient d'utiliser la requête UNION.

```
SELECT *
FROM magasin1-client
UNION
SELECT *
FROM magasin2-client ;
```

Nom	Prénom	Date-Naiss	Ville	Total
Bernard	David	27-10-1984	Metz	284
Durand	Sabrina	03-06-1996	Marseille	132
Martin	Julien	24-11-1977	Dijon	74
Daniel	François	11-04-1980	Paris	37
Pierre	Dupond	03-06-1990	Toulouse	130

Bob	Martin	10-01-1972	Lille	70

3.15. La commande INTERSECT

La commande SQL INTERSECT permet d'obtenir l'intersection des résultats de deux requêtes. Cette commande permet donc de récupérer les données communes à deux requêtes. Cela peut s'avérer utile lorsqu'il faut trouver s'il y a des données similaires sur deux tables distinctes.

La syntaxe de la commande **INTERSECT est la suivante**:

```
SELECT *
FROM table1
INTERSECT
SELECT *
FROM table2 ;
```

Pour l'utiliser convenablement il faut que les deux requêtes retournent le même nombre de colonnes, avec les mêmes types et dans le même ordre.
Exemple :
Prenons l'exemple de deux magasins qui appartiennent au même groupe. Chaque magasin possède sa table de clients.

Magasin1-Client1

Nom	Prénom	Date-Naiss	Ville	Total
Bernard	David	27-10-1984	Metz	284
Durand	Sabrina	03-06-1996	Marseille	132
Martin	Julien	24-11-1977	Dijon	74
Daniel	François	11-04-1980	Paris	37

Magasin2-Client2

Nom	Prénom	Date-Naiss	Ville	Total
Bernard	David	27-10-1984	Metz	284
Benjamaa	Reda	03-06-1990	Nante	130
Pierre	Dupond	10-01-1972	Toulouse	70
Daniel	François	11-04-1980	Paris	37

```
SELECT *
FROM magasin1-client
INTERSECT
SELECT *
FROM magasin2-client ;
```

Nom	Prénom	Date-Naiss	Ville	Total
Bernard	David	27-10-1984	Alger	284
Daniel	François	11-04-1980	Skikda	37

Le résultat présente deux tuples, il s'agit des clients qui sont à la fois dans la table « magasin1-client » et dans la table « magasin2-client ».

3.16. La commande EXCEPT/MINUS

Dans le langage SQL la commande EXCEPT s'utilise entre deux requêtes pour récupérer les données de la première requête dans inclure les résultats de la deuxième. Si une même donnée devait être présente dans les résultats des deux requêtes, ils ne seront pas présents dans le résultat final.

La syntaxe de la commande **EXCEPT/MINUS** est la suivante :

```
SELECT *
FROM table1
EXCEPT/MINUS
SELECT *
FROM table2 ;
```

Exemple : Imaginons un système informatique d'une Enterprise qui contient deux tables contenant des listes de clients.
- Une table « client-inscrits » qui contient les prénoms, noms et date d'inscription de clients.
- Une table « clients-refus-email » qui contient les informations des clients qui ne souhaitent pas être contacté par email.

Clients_inscrits

Identifiant	Nom	Prénom	Date_inscription
1	Bensalem	Mohamed	14-11-2012
2	Bendriss	Ahmed	15-2-2012
3	Diaf	Foued	17-12-2012
4	Snani	Riad	18-12-2012

Clients_refus_email

Identifiant	Nom	Prénom	Date_inscription
1	Bensalem	Mohamed	14-11-2012
2	Hadidi	Mourad	5-9-2012
3	Diaf	Foued	17-12-2012
4	Snouci	Omar	8-2-2012

Afficher uniquement le prénom et le nom des utilisateurs qui accepte de recevoir des emails informatifs.
La requête SQL à utiliser est la suivante :

```
SELECT prenom, nom
FROM clients-inscrits
EXCEPT
SELECT prenom, nom
FROM clients-refus-email ;
```

Nom	Prénom
Hadidi	Mourad
Diaf	Foued

Ce tableau montre bien les utilisateurs qui sont dans la table « clients-inscrits » et qui ne sont pas présents dans la deuxième table « clients-refus-email ».

3.17. Mettre à jour les données

Le SQL permet la modification d'une table par un utilisateur. La modification d'une table consiste à :

- ✓ Ajouter des tupes
- ✓ Modifier des tupes existants
- ✓ Supprimer des tupes.

3.17.1. Ajouter des données

L'insertion de données dans une table s'effectue à l'aide de la commande INSERT INTO. Cette commande permet au choix d'inclure un seul tuple ou un ou plusieurs tuples.

Pour insérer des données, il y a deux syntaxes principales :

- ✓ Insérer un tuple en indiquant les informations pour chaque attribut existant (en respectant l'ordre).

Syntaxe :

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

Exemple :

```
INSERT INTO client
VALUES
('Maria', 'adem', 'annaba', 16) ;
```

- ✓ Insérer un tuple en spécifiant les attributs que vous souhaitez compléter. Il est possible d'insérer un tuple en utilisant seulement une partie des attributs.

Syntaxe :

```
INSERT INTO table
(nom-colonne1, nom-colonne2, …)
Values ('valeur1', 'valaur2', …) ;
```

Exemple :
INSERT INTO client (prenom, nom, age)
VALUES
('MARIA', 'ADEM', 16) ;
Il est possible d'ajouter plusieurs tuples à une table avec une seule requête.
Voici la syntaxe à suivre :

```
INSERT INTO table (Colone1, Colone2, …..)
VALUES
(Valuer1_1, Valuer1_2,….),
(Valuer2_1, Valuer1_2,….),
(Valuer3_1, Valuer1_2,….) ;
```

```
INSERT INTO table VALUES
(Valuer1_1, Valuer1_2,….),
(Valuer2_1, Valuer1_2,….),
(Valuer3_1, Valuer1_2,….) ;
```

Exemple:

INSERT INTO Client (Nom, prénom, Adresse, Age)
VALUES
('Bensmad','Ahmed', '31 Tarek Ben Zied' ,24) ;
('Youned','Yacine', '78 Rue Amirouche', 21) ;
('Filalli', 'Mourad', 'Didouche Mourad', 18) ;

INSERT INTO Client VALUES
('Bensmad', 'Ahmed', '31 Tarek Ben Zied' ,24);
('Youned', 'Yacine', '78 Rue Amirouche', 21) ;
(Filalli, 'Mourad, Didouche Mourad', 18) ;

3.17.1.1 La commande ON DUPLICATE KEY UPDATE

L'instruction ON DUPLICATE KEY UPDATE permet de mettre à jour des données lorsqu'une donnée existe déjà dans une table. Cela permet d'avoir qu'une seule requête SQL pour effectuer selon la convenance un INSERT ou un UPDATE.
Exemple:
INSERT INTO client (num, nom, prenom)
VALUES (10, 'adem', 'maria')
ON DUPLICATE KEY UPDATE num=num+1 ;

Cette requête se traduit comme suit:

- ✓ Insérer les données num, nom et prénom avec les valeurs respectives 10, adem, maria.
- ✓ Si la clé primaire existe déjà pour ces valeurs alors seulement faire une mise à jour de num=num+1

Dans certains cas il est intéressant d'utiliser un INSERT mais de ne rien faire si la commande a déjà été inversée précédemment. Malheureusement, si la clé primaire existe déjà la requête retournera une erreur. Et s'il y a rien à mettre à jour, la commande ON DUPLICATE KEY UPDATE ne semble pas convenir.

Exemple:
INSERT INTO table (num, nom, prenom)
VALUES (10, 'adem', 'maria')
ON DUPLICATE KEY UPDATE num=num ;
Cette requête insert les données et ne produit aucune erreur si une donnée existait déjà dans la table.

3.17.2. Modifier des données

La commande UPDATE permet d'effectuer des modifications sur des tuples existantes. Très souvent cette commande est utilisée avec WHERE pour spécifier sur quels tuples doivent porter la ou les modifications.

Voici la syntaxe :

```
UPDATE table
SET nom-attribut1='nouvelle valeur'
WHERE condition ;
```

Exemple :
UPDATE client
Set age=18
WHERE nom like 'a%' ;
A noter, pour spécifier en une seule fois plusieurs modification, il faut séparer les attributions de valeur par des virgules.
UPDATE table
SET attribut1 ='valeur1', attribut2 ='valeur2', attribut3 ='valeur3'
WHERE condition ;

3.17.3 Supprimer des données

La commande DELETE en SQL permet de supprimer des tuples dans une table. En utilisant cette commande associée à WHERE il est possible de sélectionner les tuples concernés qui seront supprimées.

Syntaxe:

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

S'il n'y a pas de condition WHERE alors tout les tuples seront supprimés et la

table sera alors vide.
Exemple:

```
DELETE FROM client
WHERE age<18 ;
```

4. Langage de Définition des Données (LDD)

Cette partie présente le Langage de Définition de Données (LDD) qui permet de spécifier le schéma d'une base de données relationnelle.

4.1. Création d'une base de données

La création d'une base de données est possible en utilisant le langage SQL. Même si les systèmes de gestion de base de données (SGBD) sont souvent utilisés pour la créer.
Syntaxe :

```
CREATE DATABASE ma-base ;
```

4.2. Suppression d'une base de données

En SQL, la commande DROP DATABASE permet de supprimer totalement une base donnée et tout ce qu'elle contient. Cette commande est à utiliser avec beaucoup d'attention car elle permet de supprimer tout ce qui est inclus dans une base : les tables, les données, les vues, les index, ….
Syntaxe :

```
DROP DATA BASE ma-base ;
```

4.3. Création de tables

Une table est un ensemble de tuples et des attributs. La commande CREATE TABLE permet de créer une table en SQL. La création d'une table set à définir les attributs et le type de données qui seront contenus dans chacune des attributs (entier, chaine de caractères, date, valeur binaire,…).
Syntaxe :

```
CREATE TABLE nom-table
(
Attribut 1 type-donnees,
Attribut 2 type-donnees,
attribut 3 type-donnees,
attribut 4 type-donnees
) ;
```

Pour chaque attribut que l'on crée, il faut préciser le type de données que

l'attribut va contenir :

- **Integer, int, smallint ;**
- **Numeric (x) ;**
- **Decimal (x,y) ou numeric(x,y) ;**
- **Float (x) ;**
- **Char (x) ;**
- **Varchar (x) ;**
- **Date (aaaa-mm-jj) ;**
- **Datetime (aaa-mm-jj hh :mm :ss) ;**
- …..

Il est également possible de définir des contraintes telles que:

- **PRIMARY KEY** : Indiquer si cet attribut est considéré comme clé primaire.
- **AUTO_INCREMENT** : La valeur entière d'un attribut s'incrémente automatiquement.
- **NOT NULL** : Empêche d'enregistrer une valeur nulle pour un attribut.
- **DEFAULT** : Attribuer une valeur par défaut si aucune donnée n'est indiquée pour cet attribut lors de l'ajout d'un tuple dans la table.
- **UNIQUE** : interdit que deux tuples de la table aient la même valeur pour l'attribut.
- **CHECK** : Contrôle la validité de la valeur de l'attribut spécifié dans la condition dans le cadre d'une restriction de domaine.

Syntaxe:

```
CREAT TABLE nom-table
 (
Attribut1 type-donnees PRIMARY KEY NOT NULL
Attribut 2 type-donnees DEFAULT 'valeur'
Attribut 3 type-données
) ;
```

Exemple :

Imaginons qu'on souhaite créer une table utilisateur, dans laquelle chaque tuple correspond à un utilisateur inscrit sur un site web.

```
CREAT TABLE utilisateur (
Id-user INT PRIMARY KEY AUTO_INCREMENT,
Nom VARCHAR(25),
Prenom VARCHAR(25),
Date-naissance DATE,
Pays VARCHAR(25) DEFAULT 'Algerie',
Ville VARCHAR(50),
Code-postal VARCHAR(50),
Email VARCHAR(50),
Nombre-achat INT CHECK (nombre-achat>=0)
) ;
```

Le langage SQL permet d'indiquer quelles sont les clés étrangères dans une table, autrement dit, quels sont les attributs qui font référence à un tuple dans une autre table. On peut spécifier les clés étrangères avec l'option FOREIGN KEY.

Exemple :

```
Creat table achat
(
Id int primatry key auto_increment,
Id_user int not null,
Dateachat Date,
Montant decimal(9,2),
Foreign key (id_user) references utilisateur (id)
);
```

4.4. Modification de la structure d'une table

La commande ALTER TABLE en SQL permet de modifier une table existante. Il est ainsi possible d'ajouter un attribut, d'en supprimer une ou de modifier un attribut existant.

Syntaxe :

```
ALTER TABLE nom-table
Instruction
CREATE TABLE nom-table
) ;
```

4.4.1. Ajouter un attribut

Pour ajouter une colonne :

```
ALTER TABLE nom-table
ADD nom-attribut type-données ;
```

Exemple :

```
ALTER TABLE utilisateur
ADD adresse VARCHAR(255)
```

4.4.2. Supprimer un attribut

Syntaxe :

```
ALTER TABLE nom-table
DROP/DROP COLUMN nom-attribut ;
```

Exemple :

```
ALTER TABLE utilisateur
```

DROP code-postal ;

5.4.4.3 .Modifier le type d'un attribut

Syntaxe :

```
ALTER TABLE nom table
Modify nom-colonne type-données ;
```

Exemple :
ALTER TABLE utilisateur
MODIFY code-postal int ;

4.4.4.5 .Renommer un attribut

Syntaxe :

```
ALTER TABLE nom-table
CHANGE attribut-ancien-nom
Attribut-nouveau-nom type-données ;
```

Exemple :
ALTER TABLE utilisateur
CHANGE code-postal CP VARCHAR(5)

4.4.5 .Ajouter des contraintes d'intégrités

La modification d'une table consiste en plus des actions précédemment, d'ajouter des contraintes d'intégrité sur un attribut.

Syntaxe :

```
ALTER TABLE nom-table
Add constraint instruction ;
```

Exemple :
ALTER TABLE utilisateur
ADD constraint UNIQUE (telephone) ;

Exemple :
ALTER TABLE achat
ADD Constraint check
(montant >=100) ;

4.4.6. Renommer une table

Syntaxe :

```
ALTER TABLE table
RENAME TO new-name ;
```

Exemple:
ALTER TABLE utilisateur
RENAME TO user ;

4.4.7. Supprimer une table

La commande DROP TABLE en SQL permet de supprimer définitivement une table d'une base de données. Cela supprime en même temps les éventuels index, trigger, contraintes et permissions associées à cette table.

Syntaxe :

```
DROP TABLE nom-table ;
```

Exemple :
DROP TABLE achat

Chapitre 6
Application avec MS-ACCESS

1. Introduction

Microsoft Access est un logiciel de gestion de base de données relationnelle éditée par Microsoft. Ce logiciel fait partie de la suite Microsoft Office.
Access est un SGBDR permettant de gérer des données en masse. Il permet de créer des bases de données plus rapidement et de les gérer plus facilement. En outre, Access fournit des outils qui facilitent la communication et la collaboration. Microsoft Access offre un ensemble d'objets à savoir :

- Les tables : servent à stocker les informations. Ce sont des fichiers contenant un ensemble d'informations autour d'un même thème ou concept.
- Les requêtes : servent à filtrer les données en fonction de critères précis. Elles permettent également de réaliser des actions sur ces données (calculs, modifications, suppressions...)
- Les formulaires : permettent la saisie et la modification d'informations dans les tables de manière plus conviviale que dans les tables (intégrer des cases à cocher, des zones de texte, des images, …).
- Les ETATS : servent à imprimer les données, et permettent de présenter un même fichier de données de façons différentes.
- Les macros : permettent d'automatiser certaines actions, en programmant des boutons de commande.
- Les modules : servent à programmer de manière beaucoup plus pointue que les macros.
- Le langage SQL est le langage informatique universel qui permet de manipuler les objets et les données des bases de données.

2. Démarrage de L'Access et création d'une BDD

A partir du menu **Menu Démarrer > Tous les Programmes > Microsoft Office > Microsoft Access**
Au lancement d'Access cliquez sur le bouton **Base de données vide** :

Figure 6.1 : Ecran à l'ouverture d'Access

Dans le menu de droite, cliquer sur **Créer > Bases de données vide.**
Donc il faut choisissez alors un dossier (voire créez-en un nouveau) sur votre compte pour sauvegarder vos fichiers. Ce premier fichier s'intitulera Bibliotheque.mdb

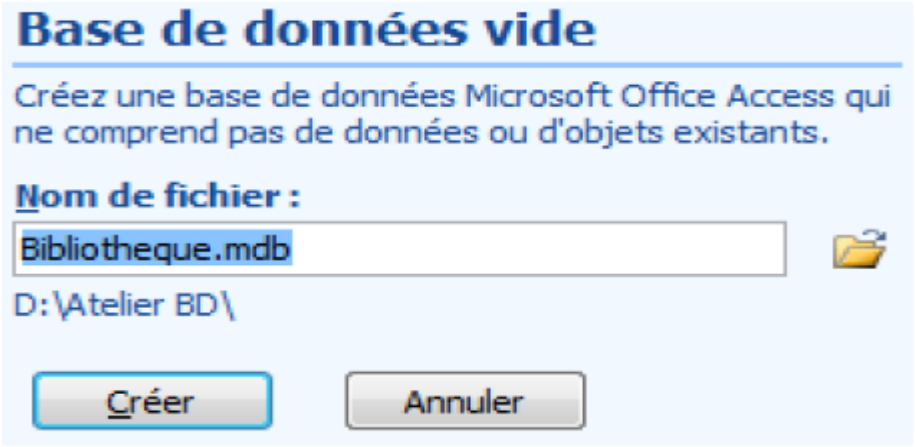


Figure 6.2 : Boîte de dialogue pour appliquer la création d'une BDD

Il faut cliquer sur Créer pour créer la base de données Exemple1.mdb, la boîte de dialogue suivante apparaît alors :

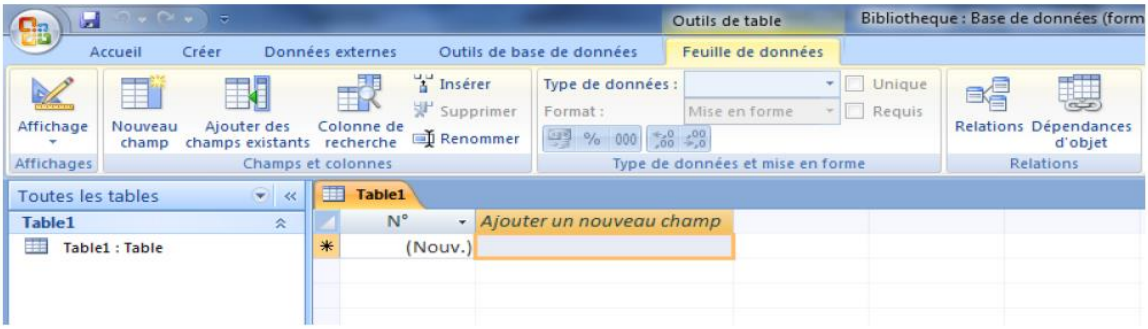


Figure 6.3 : Boîte de dialogue permettant les actions sur une BDD

3. Création d'une table dans une BDD

Cliquez sur le 1er bouton de la barre d'outils **Affichage** pour basculer du **Mode Feuille de données** au **Mode Création**. Ce dernier va vous permettre de définir le nom de la 1ère table de votre base de données puis l'intitulé et le type de vos champs (=colonnes) :

Figure 6.4 : Choix du mode création

Donnez le nom « Livre » puis validez en cliquant sur le bouton **OK** :

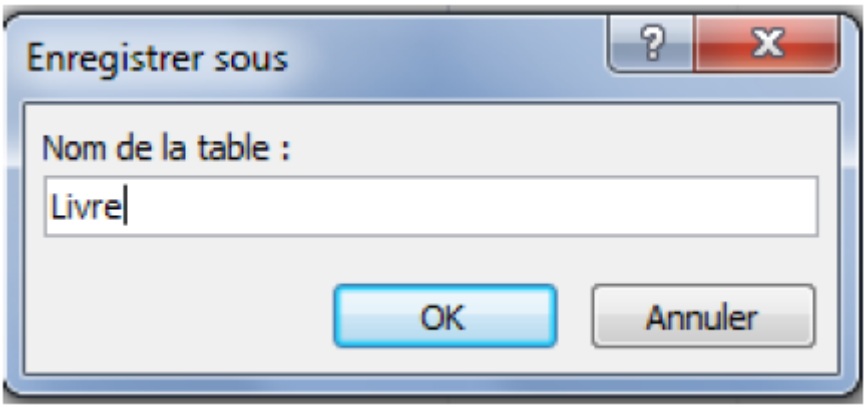


Figure 6.5 : Enregistrer table

La fenêtre suivante s'affiche :

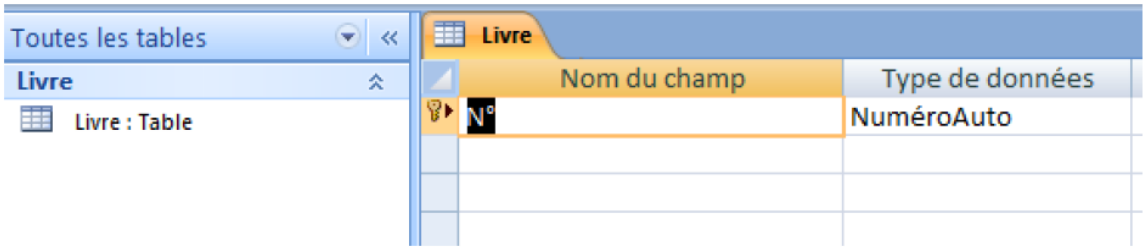


Figure 6.6 : Fenêtre de création et modification des attributs

On veut créer alors la table Livre défini comme suit :
Livre (NumInv, Matière, Titre, Auteur, Qte)
Donc il faut Cliquer : Créer (ou Create)

=> Access crée automatiquement une table en mode : Table sans assistant.

- Activer le mode création et paramétrer les champs
- Cliquer l'onglet : Créer
- Cliquer l'outil : Création de table

On se retrouve avec l'écran suivant (figure 6.7) où nous allons définir les champs de la table: nom du champ (Field Name), type de données (Data Type) du champ et, éventuellement, une description (champs Optionnel dans le lequel on peut ajouter une description concernant un attribut de la table).
Pour créer les champs de la table Livre : Saisir les champs

- Cliquer dans la zone : Nom du champ et saisir le nom du champ : NumInv,
- Zone : Type de données : Cliquer le bouton déroulant puis le type : Numérique
- Cliquer la 2eme ligne
- Cliquer dans la zone : Nom du champ et saisir le nom du champ : Matière

- Zone : Type de données : Cliquer le bouton déroulant puis le type : Texte
- Recommencer pour chaque champ de la table Livre

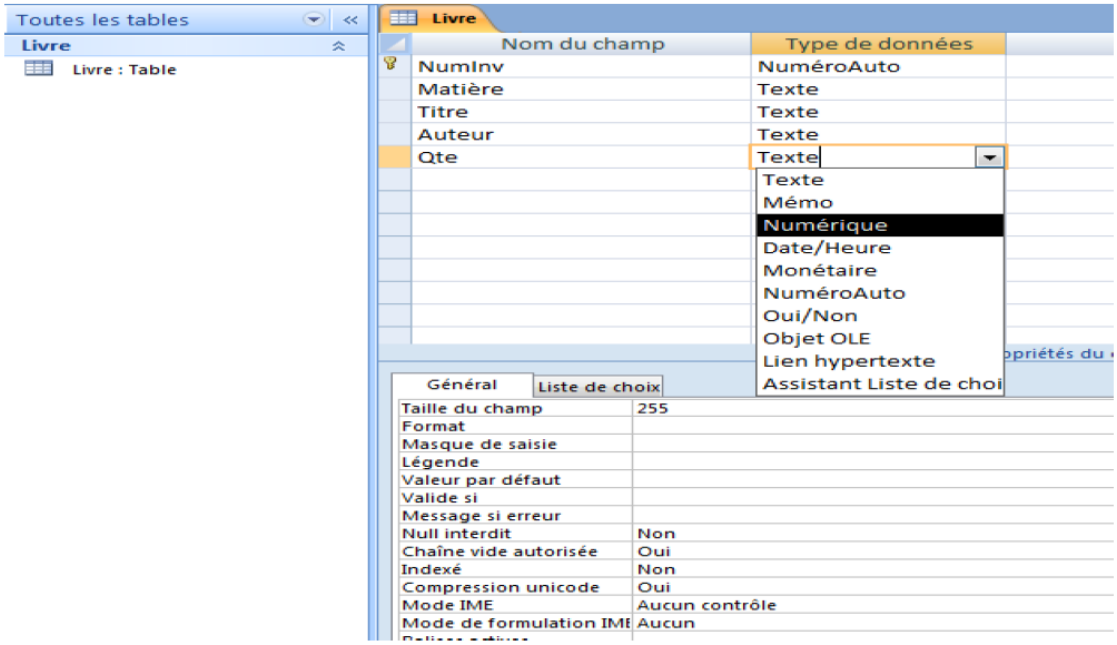


Figure 6.7 : Fenêtre de création et modification des attributs

Après l'ajout de tous les champs, il faut par la suite indiquer la clé primaire de la table.
Pour indiquer la clé primaire, il faut

- Cliquer le champ qui servira de clé primaire : NumInv
- Cliquer l'outil : Clé primaire (Primary Key)

Figure 6.8 : Fenêtre d'attribution d'une clé primaire à un champ

Repassez maintenant en **Mode Feuille de données** en cliquant sur le bouton **Affichage** dans la barre d'outils. Un message vous demande d'enregistrer votre table. Validez en cliquant sur le bouton **Oui** :

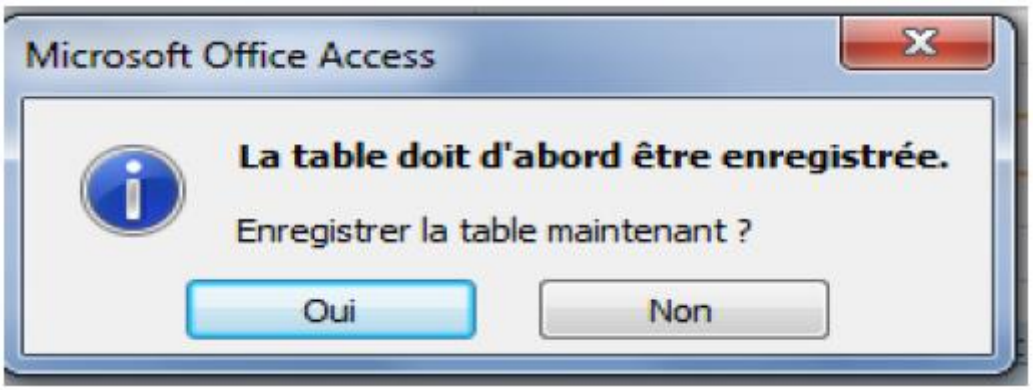


Figure 6.9 : Fenêtre d'enregistrement de la table

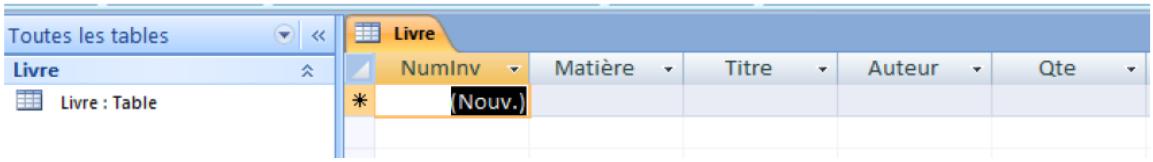


Figure 6.10 : Fenêtre d'une table Livre de la base de données

Il faut maintenant saisir des valeurs dans la table. Chaque ligne correspond à un **enregistrement**. Ne vous préoccupez pas des valeurs du champ ID elles s'incrémenteront de +1 à chaque nouvel enregistrement.

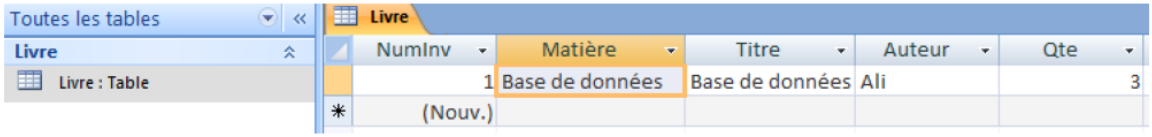


Figure 6.11 : Contenu d'une table Livre de la base de données

4. L'outil Formulaire

Pour créer le formulaire de la table Livre en utilisant l'outil Formulaire

- Activer la table ou la requête pour laquelle créer un formulaire dans le volet de navigation
- Cliquer l'outil qui correspond au formulaire à créer

5. Exploiter une base de données Access

SQL (Structured Query Language) est un langage de programmation destiné à stocker, à manipuler et à retrouver des données enregistrées dans des bases de données relationnelles.

- Ouvrir la base de données « Bibliotheque.mdb ».

Dans la barre de menus en haut cliquez sur Créer :

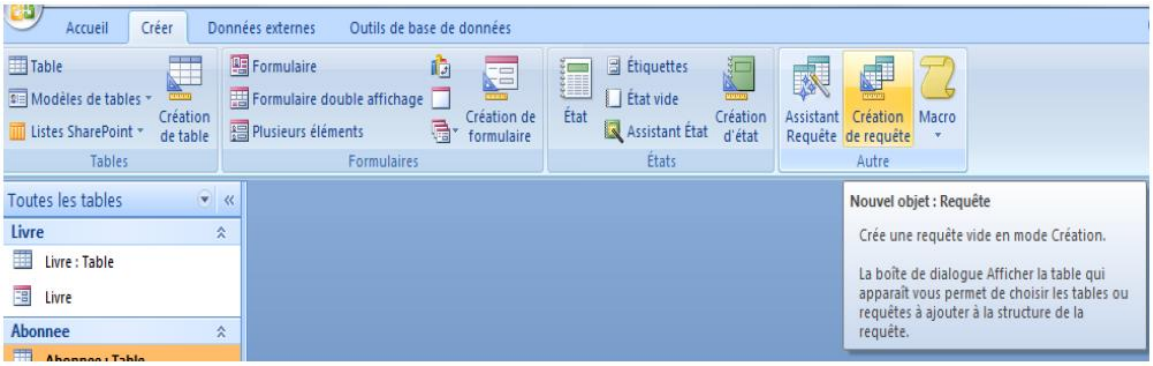


Figure 6.12 : Création d'une requête

Ensuite allez tout à droite dans la barre d'outils dans la section Autre et cliquez sur Création de requête :

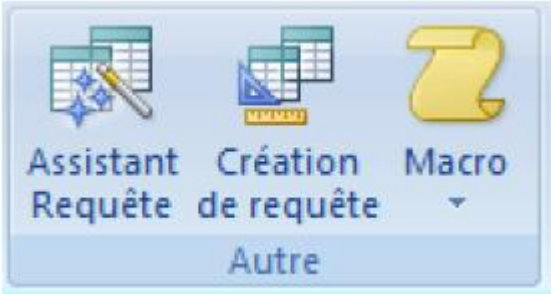


Figure 6.13 : Créer une requête

La fenêtre suivante s'ouvre avec les différentes tables de votre base de données, vous allez passer directement en mode SQL pour cela cliquez sur le bouton Fermer :

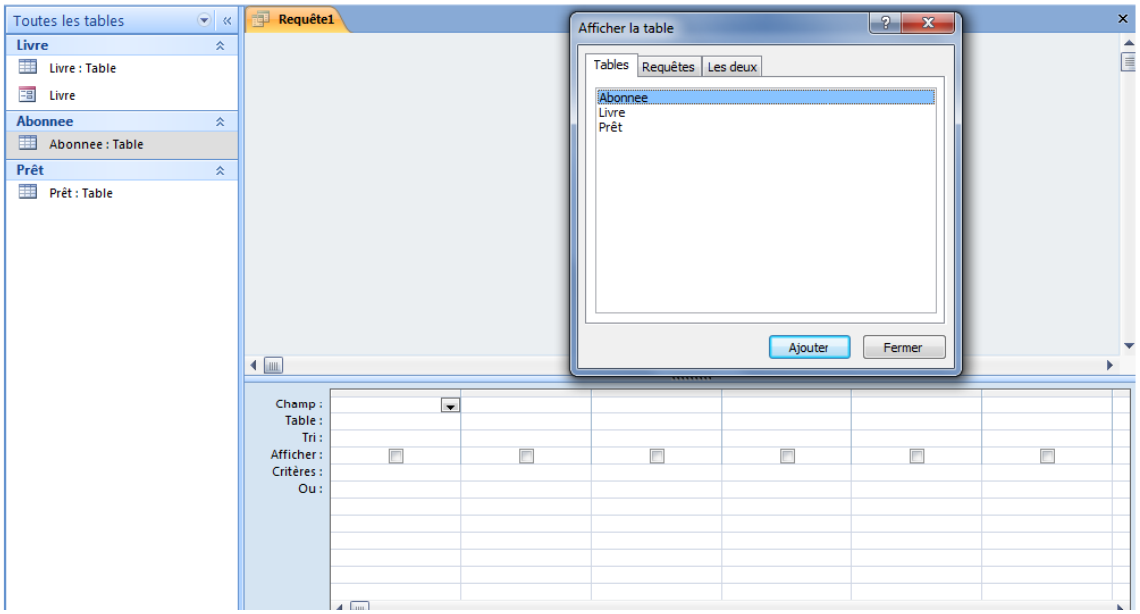


Figure 6.14 : Les différentes tables de la BDD

Cliquez maintenant tout en bas à droite sur le bouton SQL :

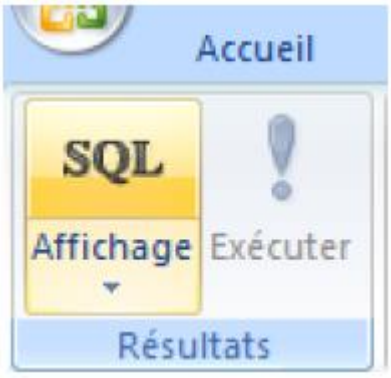


Figure 6.15 : Choisir SQL

La page principale affiche le résultat suivant :

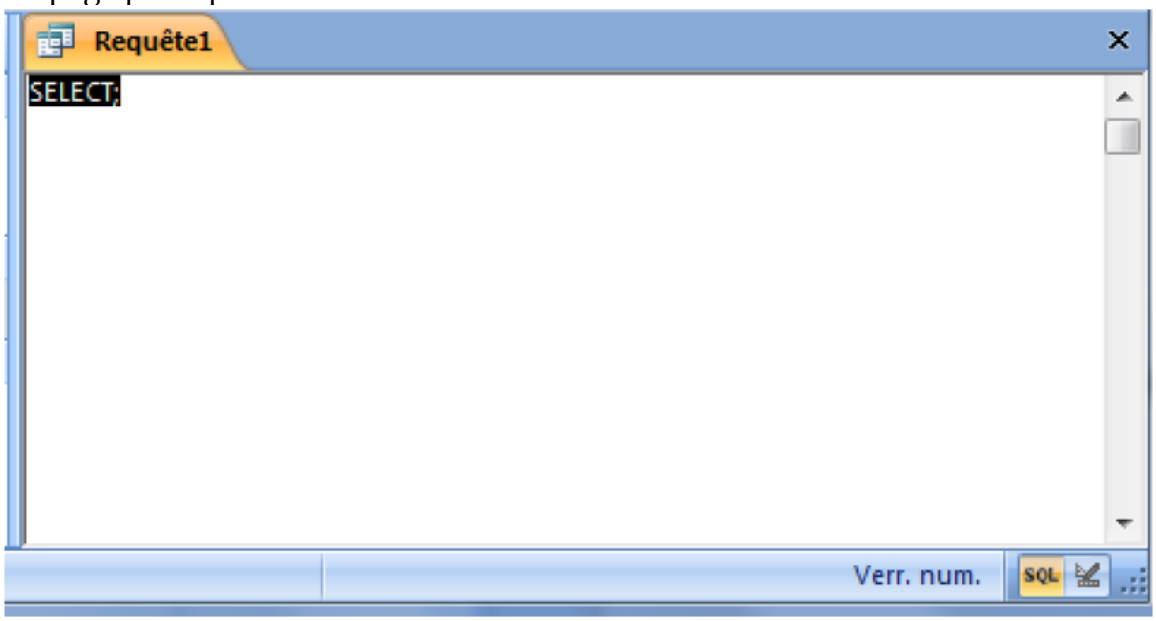


Figure 6.16 : Fenêtre pour écrire une requête

Modifiez le résultat affiché comme ci-dessous :

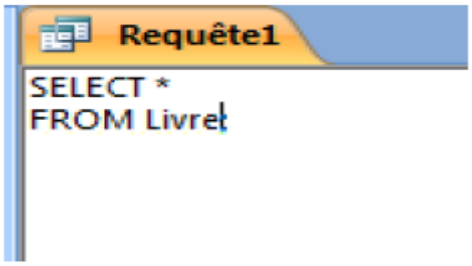


Figure 6.17 : Nouvelle requête

Cliquez ensuite sur le bouton Exécuter situé dans la barre d'outils en haut à gauche dans la section Résultats :

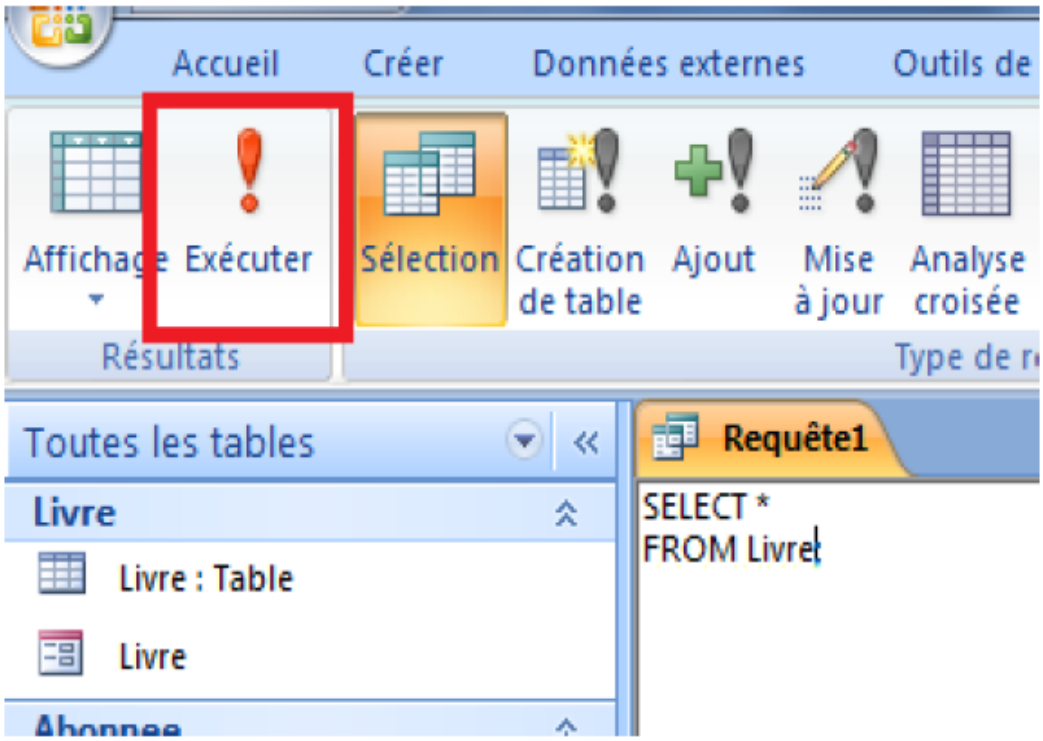
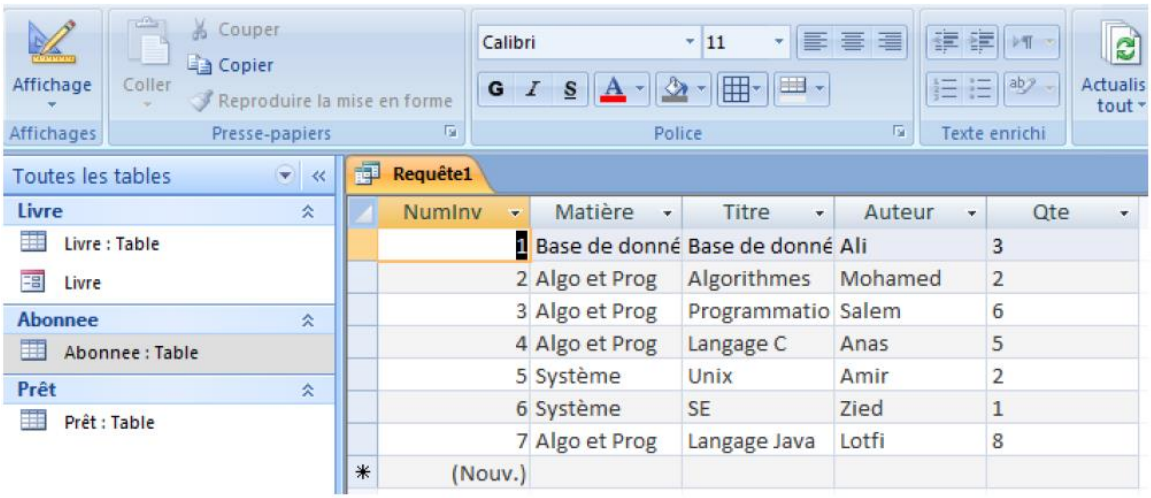


Figure 6.18 : Exécuter requête

Félicitations vous venez d'exécuter votre première requête SQL dont le résultat est le suivant :



NumInv	Matière	Titre	Auteur	Qte
1	Base de donné	Base de donné	Ali	3
2	Algo et Prog	Algorithmes	Mohamed	2
3	Algo et Prog	Programmatio	Salem	6
4	Algo et Prog	Langage C	Anas	5
5	Système	Unix	Amir	2
6	Système	SE	Zied	1
7	Algo et Prog	Langage Java	Lotfi	8
(Nouv.)				

Figure 6.19 : Résultat de la requête

Vous pouvez enregistrer la requête. Pour cela faites un clic droit sur l'onglet « Requête1 » puis un clic gauche sur Enregistrer :

Figure 6.20 : Enregistrer Requête

Une fenêtre apparait vous demandant d'enregistrer sous le nom que vous souhaitez la requête. Tapez « Liste des Amis » puis cliquez sur le bouton OK :

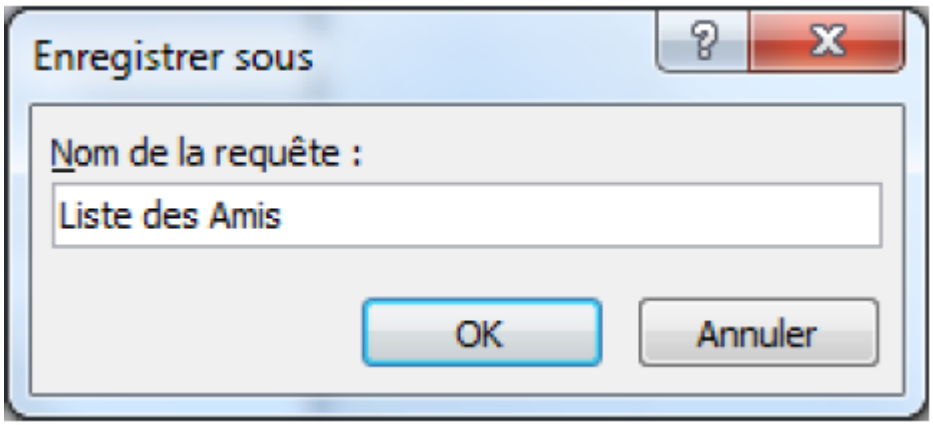


Figure 6.21 : Attribuer un nom à la requête

On constate que la requête « Liste des Amis » apparait dans la liste des tables à gauche et se distingue des tables par un symbole qui précède son nom représentant des tables liées :

Figure 6.22 : Affichage

Chapitre 7
Exercices et travaux pratiques

Exercices

Série d'exercices N°1

Exercice 1 :
Soit une liste des données recensées dans un établissement scolaire :
Nom de l'élève, Prénom de l'élève, email de l'élève, libellé matière, nombre d'heures, code filière, libellé filière, note, numéro de l'élève, numéro de la matière, langue d'enseignement.

Les règles de gestion appliquées dans cet établissement :
RG1 : chaque élève est attribué à une et une seule filière.
RG2 : une matière est enseignée pour différentes filières avec des nombres d'heures différents.
RG3 : pour tout élève, chaque matière est évaluée par une note.
Travail à faire :

1. Citer les différentes dépendances fonctionnelles.
2. Déduire le GDF associé

Exercice 2 :

Les données suivantes présentent les données rassemblées à partir d'un SI d'un établissement scolaire :
(Matricule, nom, prénom, âge, club, salle)
On considère les dépendances fonctionnelles suivantes :

Matricule →Nom, âge
Matricule → Club
Club→Salle
1. Que signifie chaque DF,
2. Proposer un modèle conceptuel en 3FN,

Exercice 3 :

On souhaite gérer un parc d'animaux, on cite les entités intervenants dans ce système ; animal, Espèce (classe d'animaux), personne, Aliment.

Exemple :

- Animal : Chat, Dauphin, serpent, cheval, …
- Espèce : Mammifère, reptile, poisson, ….
- Personne : Rachid, Yacine, Sarah
- Aliment :

En prenant en considération les règles suivant :

- ✓ Un animal appartient à une espèce et une seule ;
- ✓ Une personne peut aimer plusieurs animaux ou aucuns
- ✓ Un animal peut être aimé par plusieurs personne ou aucun
- ✓ Un animal mange au minimum un aliment
- ✓ Un aliment peut être mangé par plusieurs animaux ou aucun
- ✓ Un aliment mangé par une personne n'est pas un aliment.

Questions :

1. Etablir un modèle permettant de relier les entités ci-dessus par des associations convenables.

2. Inscrire les cardinalités sur le modèle.

Exercice 4 :

Soit la relation :
COURS (NOMPROF, VILLE, DEPARTEMENT, NOMETUDIANT, AGE, COURS, NOTE)

Soit l'ensemble des dépendances fonctionnelles initiales suivant :
1. NOMPROF → VILLE
2. VILLE → DEPARTEMENT
3. NOMPROF → DEPARTEMENT
4. NOMETUDIANT → AGE
5. NOMETUDIANT, COURS → NOTE
6. COURS → NOMPROF
Quelle est la forme normale de la relation R ? Si elle n'est pas en 3FN proposer une décomposition en 3FN.

Série d'exercices N°2

Exercice 1 :

Soit le modèle conceptuel d'une société d'assurance composé d'un ensemble d'entités.

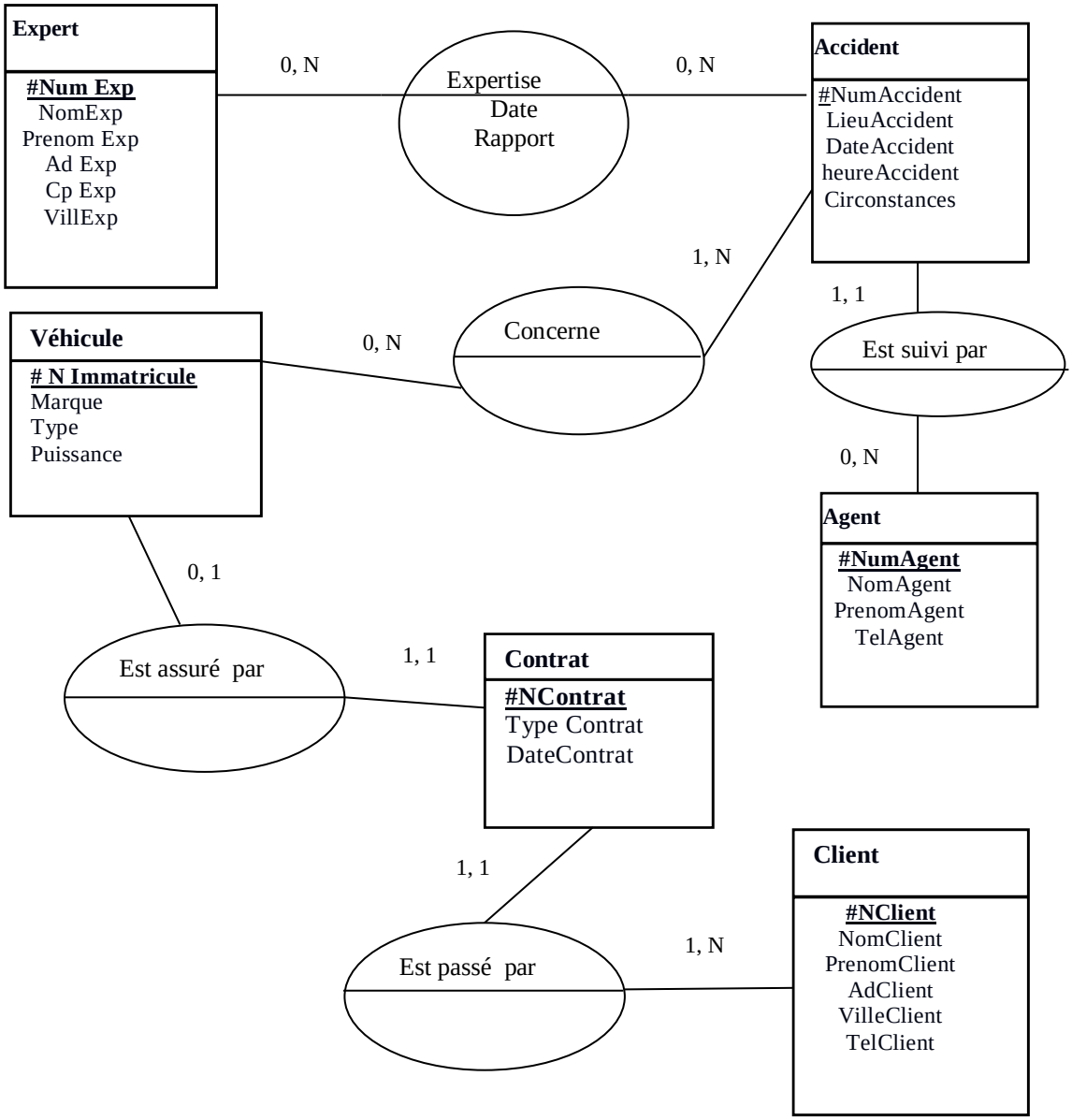
On a l'entité Expert, Accident, Agents, Clients contrats des véhicules

Un Expert expertise un Accident

Un Accident est suivi par un Agent

L'Accident concerne un Véhicule assuré par un contrat signé par un client.

Transformer le MCD en MLD.



Exercice 2 :

Une entreprise veut améliorer sa gestion du matériel ainsi que celle des commandes de ces clients. Pour cela, elle envisage les orientations suivantes :
Elle veut connaître à tout instant la quantité disponible d'un matériel dans un magasin donné. Les matériaux sont classés en catégories pour faciliter leur gestion.
On doit pouvoir connaître les composants d'un matériel et les matériaux dans lesquels on trouve un composant donné. Lors d'une rupture de stock, un matériel peut être remplacé par un matériel de substitution.
Chaque client a un seuil maximal de commandes autorisé (droit d'approvisionnement) par catégorie de matériel pour une période donnée. Un client ne peut s'approvisionner que dans un magasin et un seul. Une commande est définie par un numéro : elle concerne un seul client et différents matériaux, et précise la quantité commandée.
Q 1**.** Elaborer le modèle entités associations.

Q 2. Traduire ce modèle en un modèle relationnel.

Exercice 3 :

Un avion a un numéro d'immatriculation, un type et une localisation (la ville de l'aéroport d'attache de l'avion). Chaque type d'avion est décrit par son nom (Boeing 747, Airbus A340 ...), son poids, sa capacité et son rayon d'action. Un technicien de la compagnie a un nom, un matricule, une adresse (la ville de résidence), un numéro de téléphone, un salaire et est expert sur un ou plusieurs types d'avion pendant une période donnée (date début et date fin). Un pilote est décrit par les mêmes attributs qu'un technicien. De plus il doit passer un examen médical annuel. Chaque avion doit également passer un certain nombre de tests de bon fonctionnement. Chaque test a un numéro qui l'identifie, un nom et une valeur minimale (un seuil à atteindre). Nous souhaitons conserver la date et l'état de chacun des tests. Chaque vol est commandé par un seul pilote et concerne un seul avion. Un vol a une ville de départ (ville_dep) une ville d'arrivée (ville_arr) et une heure de départ (h_dep) une heure d'arrivée (h_arr)

Q 1. Proposer un schéma conceptuel des données (modèle entités associations). Ne pas oublier les cardinalités et de souligner les clés.
Q 2. Traduire le modèle obtenu en un modèle relationnel.

Exercice 4 :

La cuisine centrale à Montpellier voudrait gérer les données relatives à la cantine scolaire à l'aide d'une base de données relationnelle. Elle explique que le prix du repas dépend de la tranche dans laquelle l'enfant se situe et du type d'école (jardin d'enfant, maternelle, primaire). La tranche est définie en fonction du quotient familial. Chaque enfant à une carte de cantine personnelle avec un numéro. Les familles approvisionnent la carte d'un certain montant. La cuisine

centrale voudrait enregistrer tous les paiements journaliers, puis par la suite mettre à jour l'information du montant total versé. Chaque jour, elle voudrait établir et archiver une liste des enfants ayant mangé à la cantine ainsi que le menu du jour. Le menu est composé d'une entrée, d'un plat et d'un dessert.

Q 1. Elaborer le modèle entités associations.
Q 2. Traduire ce modèle en un modèle relationnel.

Série d'exercices N° 3

Exercice 1 :

Soient les schémas suivants :

Produit (idpro, désignation, marque, prix)
Vente (idcli, idpro, date, Qte)
Client (idcli, nom, ville)

Formuler les questions suivantes en langage algébrique :
1. Donner les N° des produits de marque Apple et de prix<5000da ?
2. Donner les N° des clients ayant acheté un produit de marque Apple ?
3. Donner les N° des clients n'ayant acheté que des produits de marque Apple ?
4. Donner les N° des clients ayant acheté tous les produits de marque Apple ?

Exercice 2

On considère les relations suivantes:
PERSONNE (CIN, NOM, Prenom, Adresse)
Voiture(NCarteGrise, CIN, Modele)
Moto(NCarteGrise, CIN, Modele)
Ecrire les expressions représentant:
1. Afficher les personnes qui possèdent une voiture mais pas de moto?
2. Afficher les personnes qui possèdent une voiture et une moto?
3. Afficher les personnes qui ne possèdent ni voiture ni moto?

Exercice 3

Soit le schéma de la base de données Bibliothèque suivante :
Etudiant(**NumEtd**,NomEtd,PrenomEdt,AdresseEtd)
Livre(**NumLivre**,TitreLivre,*NumAuteur*,*NumEditeur*,*NumTheme*,AnneeEdition)
Auteur(**NumAuteur**,NomAuteur,AdresseAuteur)
Editeur(**NumEditeur**,NomEditeur,AdresseEditeur)
Theme(**NumTheme**,IntituléTheme)

Prêt(**NumEtd**,**NumLivre**,**DatePret**,DateRetour) En **gras** les clés primaires et en *italique* les clés étrangères Ecrire en langage algébrique les requêtes suivantes :

1. Le nom, le prénom et l'adresse de l'étudiant de nom 'Alami'
2. Le numéro de l'auteur 'Alami'
3. la liste des livres de l'auteur numéro 121
4. les livres de l'auteur nom 'Alami'
5. le numéro de l'auteur du livre 'comment avoir 20 en BDD'
6. le nom et l'adresse de l'auteur du livre 'comment avoir 20 en BDD'
7. Les livres de l'auteur 'Alami' «édités chez l'éditeur 'Nul part'
8. Les livres de l'auteur 'Alami' ou 'Belhadj'
9. les livres qui n'ont jamais été empruntés

Exercice 4 : On considère la base de données BDD AIRBASE suivante :

PILOTE (NUMPIL : D_NUMPIL, NOMPIL: D_NOMPIL, ADR : D_VILLE, SAL : D_SAL)
AVION (NUMAV : D_NUMAV, NOMAV : D_NOMAV, CAP : D_CAP, LOC : D_VILLE)
VOL (NUMVOL : D_NUMVOL, NUMPIL : D_NUMPIL, NUMAV : D_NUMAV, VILLE_DEP : D_VILLE, VILLE_ARR : D_VILLE, H_DEP : D_HEURE, H_ARR : D_HEURE)

Q1 : Donnez la liste des avions dont la capacité est supérieure à 350 passagers.
Q2 : Quels sont les numéros et noms des avions localisés à Alger ?
Q3 : Quels sont les numéros des pilotes en service et les villes de départ de leurs vols ?
Q4 : Donnez toutes les informations sur les pilotes de la compagnie.
Q5 : Quel est le nom des pilotes domiciliés à Paris dont le salaire est supérieur à 15000 F ?
Q6 : Quels sont les avions (numéro et nom) localisés à Alger ou dont la capacité est inférieure à 350 passagers ?
Q7 : Liste des vols au départ de Constantine allant à Alger après 18 heures ?
Q8 : Quels sont les numéros des pilotes qui ne sont pas en service ?
Q9 : Quels sont les vols (numéro, ville de départ) effectués par les pilotes de numéro 100 et 204 ?
Q10 : Donnez le numéro des vols effectués au départ d'Alger par des pilotes Algérois ?
Q11 : Quels sont les vols effectués par un avion qui n'est pas localisé à Alger ?
Q12 : Quels sont les pilotes (numéro et nom) assurant au moins un vol au départ de Constantine avec un avion de capacité supérieure à 300 places ?
Q13 : Quels sont les noms des pilotes domiciliés à Alger assurant un vol au départ de Constantine avec un Airbus ?
Q14 : Quels sont les numéros des vols effectués par un pilote Algérois au départ ou à l'arrivée d'Alger avec un avion localisé à Constantine ?
Q15 : Quels sont les pilotes (numéro et nom) habitant dans la même ville que le pilote Bendali ?

Q16 : Quels sont les numéros des pilotes en service différents de celui de Bendali ?
Q17 : Quelles sont les villes desservies à partir de la ville d'arrivée d'un vol au départ d'Alger ?
Q18 : Quels sont les appareils (leur numéro) localisés dans la même ville que l'avion numéro 100 ?
Q19 : Quels sont les numéros et noms des pilotes domiciliés dans la même ville que le pilote Dupont et dont le salaire est supérieur à celui de Bendali ?
Q20 : Quels sont les numéros et noms des pilotes qui effectuent un vol au départ de leur ville de résidence ?
Q21 : Y a-t-il des homonymes parmi les pilotes ? Si oui, donner leur numéro et nom.

Série d'exercices N° 4

Exercice 1 :
Soit la relation Voiture (NIMM, marque, modèle, prix, couleur, achat)

Un extrait de la table voiture est le suivant :

Voiture

MIMM	Marque	Modèle	Prix	Couleur	Date achat
124870A15	Renault	Laguna	30	Blanc	12/07/2011
152140A15	Renault	Clio	15	Rouge	12/05/2010
156020B10	Peugeot	308	24	GriSouris	06/08/2012
156879A12	Peugeot	308	24	GrisClair	01/09/2014
168790D12	Hyundai	I30	23	Blanc	01/02/2015
169020A17	Hyundai	I30	21	GrisClair	02/05/2012
192020A15	Renault	Megane	26	Blanc	01/10/2013
196020A17	Peugeot	207	21	GrisClair	01/08/2010
254001B15	Renault	Clio	18	Noir	25/05/2012

Réaliser les requêtes suivantes :
Q1. Afficher la liste des voitures
Q2. Afficher le prix et le modèle des voitures de marque « Renault »
Q3. Afficher le modèle, le prix et le prix TTC des voitures de marque « Renault » de prix HT inférieur ou égale à 25 millions.
Q4. Afficher les marques des voitures, en éliminant les doublons au niveau d'affichage.
Q5. Afficher les différentes marques et modèles des voitures
Q6. Afficher les NIMM, marque et modèle des voitures dans l'ordre de la date d'achat.
Q6. Afficher les marques, modèle, couleur et prix des voitures par ordre croissant de marque et décroissant de prix.
Q7. Afficher le nombre de voitures de marque « Peugeot ».
Q8. Afficher le prix minimal et le prix maximal des voitures.

Q9. Afficher le nombre de voitures de chaque marque.
Q10. Modifier la requête précédente pour afficher le nombre de voitures de chaque marque ayant un prix minimale inférieur à 20 millions.

Exercice 2 :
Soit le schéma relationnel suivant :

Matière (Mcode, Titre, niveau)
VolumH (Mcode, Type, Nbh). Un extrait de ces tables est le suivant :

Matières

Mcode	Titre	Niveau
CP2	Comptabilité	2
FR1	FrançaisEcrit	1
IF1	InformatiqueI	1
IF3	InformatiqueII	2
MT1	MathEco	2

Réaliser les requêtes suivantes :

Q1. Donnez pour chaque matière: le code, le titre, le niveau et les volumes Horaire (cours, td et/ou TP)

Q2. Donnez le code des matières réalisées en cours et aussi en TP.

VolumesH

Mcode	Type	Nbh
CP2	Cours	30
CP2	TD	15
IF1	Cours	20
IF1	TP	25
IF3	Cours	15
IF3	TD	15
IF3	TP	15

Exercice 3

Soit la base de données relationnelle des vols quotidiens d'une compagnie aérienne qui contient les tables Avion, Pilote et Vol.
Table **Avion** (**NA** : numéro avion de type entier (clé primaire),
Nom : nom avion de type texte (12),
Capacite : capacité avion de type entier,
Localite : ville de localité de l'avion de type texte (10)
)

Table **Pilote** (**NP** : numéro pilote de type entier,
Nom : nom du pilote de type texte (25),
Adresse : adresse du pilote de type texte (40)
)

Table **Vol** (**NV** : numéro de vol de type texte (6),
NP : numéro de pilote de type entier,
NA : numéro avion de type entier,
VD : ville de départ de type texte (10),
VA : ville d'arrivée de type texte (10),
HD : heure de départ de type entier,
HA : heure d'arrivée de type entier
)

1) Insérer les avions suivants dans la table Avion :
(100, AIRBUS, 300, RABAT), (101,B737,250,CASA), (101, B737,220,RABAT)
2) Afficher tous les avions
3) Afficher tous les avions par ordre croissant sur le nom
4) Afficher les noms et les capacités des avions
5) Afficher les localités des avions sans redondance
6) Afficher les avions dans la localité et Rabat ou Casa
7) Modifier la capacité de l'avion numéro 101, la nouvelle capacité et 220
8) Supprimer les avions dans la capacité et inférieure à 200
9) Afficher la capacité maximale, minimale, moyenne des avions
10) Afficher les données des avions dont la capacité et la plus basse
11) Afficher les données des avions dont la capacité et supérieure à la capacité moyenne
12) Afficher le nom et l'adresse des pilotes assurant les vols IT100 et IT104
13) Afficher les numéros des pilotes qui sont en service
14) Afficher les numéros des pilotes qui ne sont pas en service
15) Afficher les noms des pilotes qui conduisent un AIRBUS

Exercice 4 :

Soit le modèle relationnel suivant relatif à la gestion des notes annuelles d'une promotion d'étudiants:

- ETUDIANT(N°Etudiant, Nom, Prénom)
- MATIERE(CodeMat, LibelléMat, CoeffMat)
- EVALUER(#N°Etudiant, #CodeMat, Date, Note)

Remarque : les clés primaires sont soulignées et les clés étrangères sont marquées par #. Exprimez en SQL les requêtes suivantes :
1. Quel est le nombre total d'étudiants ?
2. Quelles sont, parmi l'ensemble des notes, la note la plus haute et la note la plus basse ?
3. Quelles sont les moyennes de chaque étudiant dans chacune des matières ?

4. Quelles sont les moyennes par matière ?
5. Quelle est la moyenne générale de chaque étudiant ?
6. Quelle est la moyenne générale de la promotion ?
7. Quels sont les étudiants qui ont une moyenne générale supérieure ou égale à la moyenne générale de la promotion ?

Exercice 5 :

Ci-après, on donne la représentation textuelle simplifiée d'une base de données concernant un cycle de formation destiné à des étudiants. Il regroupe un ensemble de matières. On considère que chaque enseignant n'enseigne qu'une seule matière et qu'à la fin du cycle de formation, une note par matière, est attribuée à chaque étudiant. D'autre part, les étudiants peuvent ne pas suivre les mêmes matières.

- ETUDIANT(CodeEt, NomEt, DatnEt)
- MATIERE(CodeMat, NomMat, CoefMat)
- ENSEIGNANT(CodeEns, NomEns, GradeEns, #CodeMat)
- NOTE(#CodeEt, #CodeMat, note)

Exercice 6 :

Ecrire les requêtes SQL permettant d'afficher :
1. Les informations relatives aux étudiants (Code, Nom et Date de naissance) selon l'ordre alphabétique croisant du nom
2. Les noms et les grades des enseignants de la matière dont le nom est 'BDD'.
3. La liste distincte formée des noms et les coefficients des différentes matières qui sont enseignées par des enseignants de grade 'Grd3'.
4. La liste des matières (Nom et Coefficient) qui sont suivies par l'étudiant de code 'Et321'.
5. Le nombre d'enseignants de la matière dont le nom est 'Informatique'

Travaux Pratiques

TP N° 1 :

La Base de données « Bibliotheque » utilisée dans ce TP est constituée des tables suivantes :
Table des livres (Numéro inventaire du livre, matière, titre et auteur du livre, nombre d'exemplaires)

Livre :

NumInv	Matière	Titre	Auteur	Qte
1	Base de données	Base de données	Ali	3
2	Algo et programmation	Algorithmes	Mohamed	2
3	Algo et programmation	Programmation	Salem	6
4	Algo et programmation	Langage C	Anas	5
5	Système	Unix	Amir	2
6	Système	Système	Zied	1

		d'exploitation		
7	**Algo et programmation**	**Langage Java**	**Lotfi**	**8**

- Table des abonnés (Numéro d'abonné, son nom, son prénom, son statut et le département d'affectation)

Abonne :

NumAB	Nom	Prénom	Département
23	Julien	Martin	TI
24	David	Bernard	GM
25	Marie	Leroy	TI
26	Pierre	Dupond	TI

- Table des prêts (Numéro inventaire du livre emprunté, Numéro d'abonné de l'emprunteur, la date d'emprunt, la date de retour et une observation)

Prêt :

NumAb	NumInv	Date Pret	DateRetour	Observation
23	4	01/09/2015	07/09/2015	Il ne faut pas dépasser 8 jours
26	1	02/10/2015	15/10/2015	
26	3	03/11/2015	03/12/2015	Suppression pour un mois
24	1	16/11/2015	14/11/2015	
23	6	21/11/2015		

Question 1.
Créer les tables ci-dessus dans la même base de données (avec les clés primaires, le saisi des attributs, les types de données etc.)

Question.2
Créer le Modèle Conceptuel de données (MCD) avec ces tables (entités) de la base de données

Question.3
Saisir les données des trois tables.

TP N° 2 :

1) Créer une nouvelle base de données dans le dossier C:\TP2 et lui donner le nom « Gestion
des notes ».
2) Créer les tables suivantes en respectant le schéma de la base de données présenté ci-
dessous
ELEVE (ID_ELEVE, Nom, Prénom, Date_naissance, Classe)
MATIERE (ID_MATIERE, libellé_matière, coefficient)
NOTE (ID_ELEVE, ID_MATIERE, Note)

La table Elève (La clé primaire est ID_ELEVE)

Champ	Type de données	Propriétés

Id-Elève	Texte	Taille = 4
Nom	Texte	Taille = 20
Prénom	Texte	Taille = 20
Date_naissance	Date/Heure	Format abrégé
Classe	Texte	Taille =8

La table Matière (La clé primaire est ID_MATIERE)

Champ	Type de données	Propriétés
Id_Matière	Texte	Taille = 3
Libellé_ Matière	Texte	Taille = 20
Coefficient	Numérique	Taille = réel simple

La table Note (La clé primaire est ID_ELEVE, ID_MATIERE)

Champ	Type de données	Propriétés
Id_Elève	Texte	Taille = 4
Id_Matière	Texte	Taille = 3
Note	Numérique	Taille = réel simple

3) Créer les relations possibles entre les tables de la base de données « Gestion des notes »
4) Remplir chacune des tables de la base de données par les données correspondantes :

- La table ELEVE

Champ	**Nom**	**Prénom**	**Date de naissance**	**Classe**
E001	Durand	Sabrina	18/09/1992	4EC01
E002	Martin	Julien	23/10/1991	4 EC02
E003	Francois	Daniel	22/01/1992	4 EC03

- La table MATIERE

ID_Matière	**Libéllé _Matière**	**Coefficient**
E001	Economie	18/09/1992
E002	Gestion	23/10/1991
E003	Philosophie	22/01/1992

- La table NOTE

ID-Elève	**ID-Matière**	**Note**
E001	M01	14
E002	M02	13
E003	M03	10
E004	M03	11
E005	M01	17
E006	M02	14,5

5) Réaliser les requêtes suivantes :

- R1 : Afficher la liste de tous les élèves
- R2 : Afficher la liste des élèves de la classe « 4ECO1 »
- R3 : Afficher la liste des matières dont le coefficient est supérieur ou égale à 2
- R4 : Afficher la liste des élèves dont le nom commence par la lettre « B »
- R5 : Afficher les noms et les prénoms des élèves ayants une note supérieure à 15
- R6 : Afficher les noms et prénoms des élèves ayant une note supérieure à 10 en philosophie

TP N° 3 :

1. Création d'une base de données

Créer la base de données « Vente » de schéma relationnel suivant :

Magasin (num_m, loc)
Frs (num_f, nom, ville)
Client (num_c, nom, pays, ville)
Article (num_a, des, poids, couleur, prix_achat, num_f)
Vente (num_c, num_a, num_m, qte, prix_vente, dat)

Avec :

-Les attributs num_m, loc, num_f, nom, ville, num_c, nom, pays, ville, num_a, des, couleur sont de type chaine de caractère de longueur maximale 30.

-L'attribut poids est un entier de longueur maximale 20.

-Les attributs prix_achat, qte, prix_vente sont des réels de taille 10 chiffres au total et 6 chiffres avant la virgule.

-L'attribut dat est de type DATE.

2. Mise à jour de la structure d'une table

Modifier les tables crées en ajoutant les contraintes présentés dans le schéma de base de données affichés ci-dessous :

Magasin (num_m, loc)
Frs (num_f, nom, ville)
Client (num_c, nom, pays, ville)
Article (num_a, des, poids, couleur, prix_achat, # num_f)
Vente (#num_c, #num_a, #num_m, qte, prix_vente, dat)

Avec :

-les attributs soulignés sont des clés primaires.

-Les attributs précédés par le caractère « # » sont des clés étrangères.

TP N° 4 :

Soit la base de données FABRICATION dont le schéma est donnée ci-dessous.
PIECE (NOP, Désignation, Couleur, Poids)
SERVICE (NOS, Intitulé, Localisation)
COMMANDE (NOP, NOS, Quantité)

1) Créer la base de données FARICATION et l'enregistrer dans votre dossier de travail situé sous le dossier TP4 du racine C: puis créer les tables de cette base de données ainsi que les relations qui les relient.

2) Remplir les tables par les données suivantes.

Service

NOS	Intitulé	Localisation
1	S1	LOC1
2	S2	LOC2
3	S3	LOC3
4	S4	LOC4

Pièce

NOP	Designation	Couleur	Poids
11	P1	Rouge	5,3
12	P2	Vert	26,22
13	P3	Bleu	13
14	P4	Rouge	25
15	P5	Gris	17,75

Commande

NOS	Intitulé	Localisation
1	S1	LOC1
2	S2	LOC2
3	S3	LOC3
4	S4	LOC4

3) Créer les requêtes suivantes :
   a. Les noms des services ayant en commande la pièce « P1 » avec une quantité supérieure
   à 10, dans l'ordre croissant.
   b. La liste des pièces commandées par un service donné.
   c. L'augmentation de 50% des quantités des pièces commandées par le service S1.
4) Créer les formulaires suivants :
   a. Un formulaire de saisie des Services.
   b. Un formulaire de saisie des Pièce.
   c. Un formulaire de saisie des commandes.
   d. Un menu général : contenant le titre « Gestion fabrication », les boutons de commandes :
   - Le 1er pour afficher le formulaire de saisie des services.
   - Le 2éme pour exécuter la première requête.
   - Le 3éme pour imprimer la table commande.
   - Le 4éme pour quitter le formulaire en cour.
5) Ajouter dans le menu les boutons de commandes suivants :
   - Le bouton de commande « Saisie des pièces » qui, sur clic, ouvre le formulaire « saisie des

pièces » à l'aide d'une macro.
- Le bouton de commande « Saisie des commandes » qui, sur clic, ouvre le formulaire « saisie des commandes » à l'aide d'une macro.
- Le bouton de commande « requête 2 » qui, sur clic, ouvre le formulaire « requête 2» à l'aide d'une macro.

6) les états :
a. Créer l'état qui permet d'imprimer la liste des services.
b. Créer l'état qui permet d'imprimer la liste des commandes trié par ordre croissant
selon les quantités.
c. Créer l'état qui permet d'imprimer la liste des pièces commandé par un service donné.
d. Ajouter le champ calcule prix-pièce =[NOP]*[quantité]

Bibliographie

- Clouse.M . Algèbre relationnelle : Guide pratique de conception d'une base de données relationnelle normalisée, 2008. Editions ENI.

-Mbuyi Mikendi .E. Cours de Systèmes d'information et de base des données .Troisième graduat informatique 2013-2014. Tome I. Université de Kinshasa.

- Bellan.P. Introduction aux bases de données Access, 500 pages, 2006, ISBN. 978-2-7298-2577-5.

- Clemente.P. Cours de Bases de données, SQL (Structured Query Langage). Filière STI 2ème année. 2003-2004. ENSI Bourges.

- Clemente.P .Cours de Bases de données, Normalisation d'un schéma relationnel, Clemente. P, Filière STI 2ème année. 2003-2004. ENSI Bourges.

- Date. C. J. Introduction aux bases de données. 2004. Vuibert.

- Gardarin. G. Bases de données. Editions Eyrolles, 2003.

- Mbuyimukendi.E. Cours de Systèmes d'information et de base des données Troisième graduat informatique. Université de Kinshasa. 2013-2014.TomeI.

Sites web:

Https://laurent-audibert.developpez.com/Cours-BD/

Https://docplayer.fr/629070-Cours-de-bases-de-donnees-philippe-rigaux.html

Https://eddirasa.com/wp-content/uploads/univ/ESI/esi structured_query_language.pdf

Https://eddirasa.com/wp-content/uploads/univ/ESI/esi-le_langage_algebrique.pdf

Https://eddirasa.com/wp-content/uploads/univ/ESI/esi-le_model_relationnel.pdf

Https://sgbd.developpez.com/tutoriels/cours-complet-bdd-sql

Https://www.univ-orleans.fr/lifo/Members/duchier/teaching/A2-OMGL SQL/CoursSQL2.pdf

Https://www.i3s.unice.fr/~nlt/cours/licence/sgbd1/sgbd1_cours.pdf

Http://developpement-informatique.com/cours/dev-info/langage-SQL/45/Exercices-corriges-de-langage-SQL

Http://www.livrespourtous.com/Livres-bases-de-donnees.html

Https://www.doc-etudiant.fr/Informatique/Bdd/Cours-Exercices-corriges-de-conception-de-base-de-donnees-9763.html

Http://www.ybet.be/access/

Http://www.cuy.be/cours/access/ex_qr_ques.html