REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université TAHRI Mohammed Béchar
Faculté des Sciences Exactes
Département des Mathématique et Informatique

N° d'ordre :

Filière : Informatique

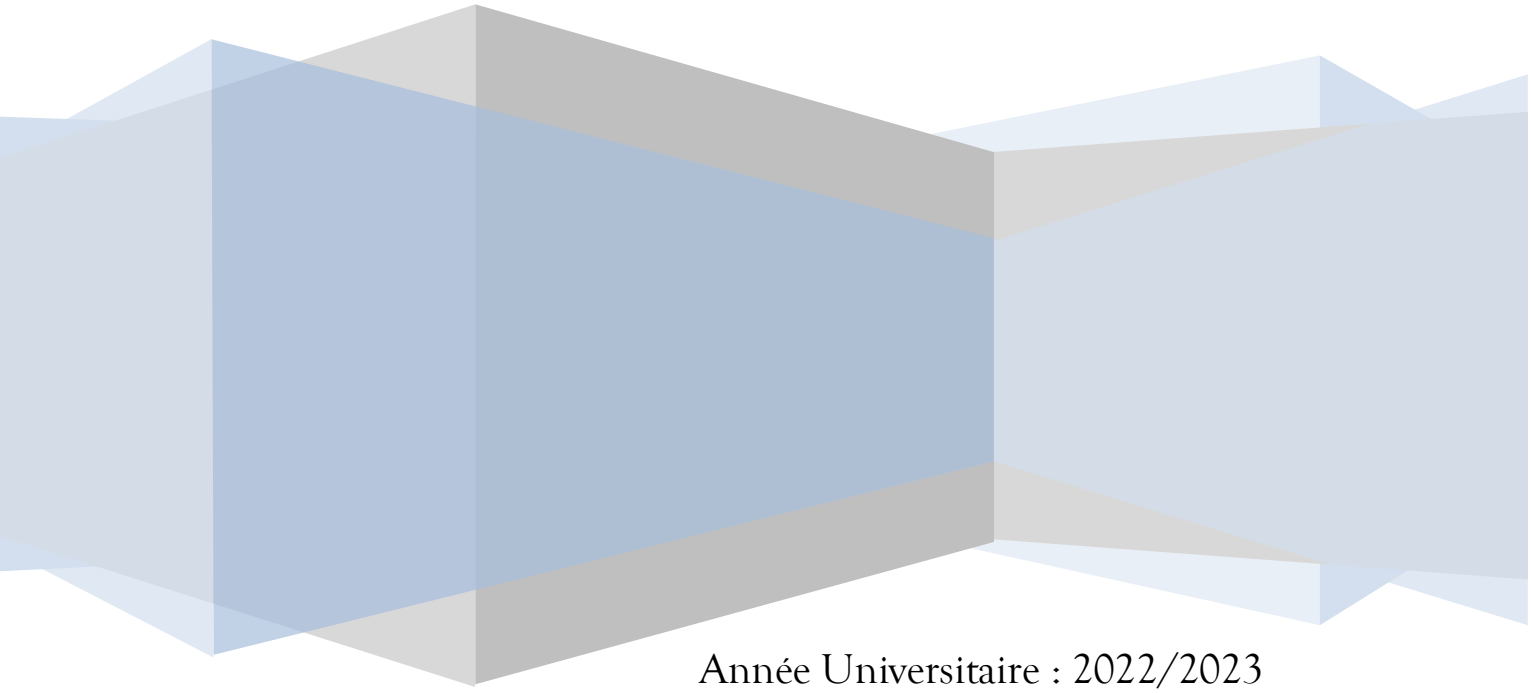
Spécialité : Licence : Systèmes Informatiques

Module : SGBD

Système de Gestion de Base de Données (SGBD)

Cours & Exercices

BENDJIMA Mostefa



Année Universitaire : 2022/2023

Avant propos

Ce cours vise à introduire les principes des bases de données informatiques aux étudiants de deuxième année licence en informatique au programme national. Après la première année licence où les étudiants auraient appris les bases de l'informatique ainsi que les fondements de la programmation, ce cours constitue une continuité de la base de la programmation informatique qui traite la gestion des données dans les systèmes.

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 analyser l'univers d'application ;
- 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.

Table des matières

Table des figures

I. Introduction

I.1. Historique des SGBD

Les SGBD ont déjà une longue histoire. Les années 60 ont connu un premier développement des systèmes de fichiers qui tendent essentiellement à faire apparaître les mémoires secondaires comme idéales, partagées, banalisées, directement adressables, de capacité infinie. Ceux-ci, après avoir évolué, composent aujourd'hui le cœur des SGBD.

Le milieu des années 60 a vu la naissance de la première génération de SGBD ; celle-ci a été marquée par la séparation de la description des données des programmes d'application et l'avènement de langages d'accès navigationnels, c'est-à-dire permettant de se déplacer dans des structures de type graphe et d'obtenir, un par un, des groupes de données appelés articles. Cette première génération, caractérisée par les systèmes obéissent aux premières recommandations du CODASYL est basée sur des modèles d'accès, c'est à dire des modèles de données visant essentiellement à optimiser les méthodes de placement des données sur les mémoires secondaires afin de réduire les temps d'accès [1].

La deuxième génération de SGBD a grandi dans les laboratoires depuis 1970, à partir du modèle relationnel. Elle vise essentiellement à enrichir le SGBD externe afin de simplifier l'accès aux données pour les utilisateurs externes. Ainsi, la deuxième génération offre des langages assertionnels basés sur la logique, permettant de spécifier les données que l'on souhaite obtenir sans dire comment les accéder [1].

La troisième génération de SGBD basée sur des langages d'accès plus puissants et plus naturels, supportant des types de données plus variés, incluant des possibilités de déduction et de répartition de données [1].

Ce chapitre a pour objet de définir les concepts de base de données (BDD) et de système de gestion de base de données (SGBD).

I.2. Concept de base de données

Une base de données est faite pour enregistrer des faits, des événements qui surviennent dans la vie d'un organisme et pour les restituer à la demande ou bien pour tirer des conclusions en rapprochant plusieurs faits élémentaires les uns aux autres [1,2].

Par exemple, une université a besoin pour sa gestion d'archiver des informations sur les enseignants, les étudiants, les enseignements, les conditions d'inscriptions à une formation…. etc. Mais elle a aussi besoin de connaître les associations qui existent entre ces différentes informations : l'inscription qui associe l'étudiant avec la formation qu'il souhaite, les modules donnés par un enseignant…etc.

Vu sous l'angle des anciennes méthodologies de conception, chaque volet de ce cas d'exemple nécessiterait une nouvelle application avec ses propres fichiers et ses propres programmes. La création d'une base de données va à l'encontre de cette façon de faire : elle rend possible la centralisation, la coordination, l'intégration et la diffusion de l'information archivée.

D'une manière générale, on peut définir une BDD de cette manière :

Une base de données est un ensemble structuré de données enregistrées sur des supports de stockage pour satisfaire simultanément plusieurs utilisateurs de façon sélective en un temps opportun [3].

Une base de données est construite selon un modèle de données, il existe trois grandes catégories de modèles de données : le modèle réseau, le modèle hiérarchique et le modèle relationnel.

I.3. Notion de SGBD

Un SGBD est un ensemble de logiciels et matériels associés permettant aux utilisateurs d'interagir avec une BDD [3]. Il permet principalement de créer, d'insérer, de modifier et de rechercher efficacement des données spécifiques dans une grande masse d'informations partagées par tous les usagers. Pour aboutir à ce résultat, l'utilisateur décrit en termes abstraits ce qu'il souhaite faire sur les données, laissant le soin au système d'effectuer les tâches de recherche en fonction de la représentation et de l'organisation des données sur les supports physiques.

Par exemple, vous arrivez dans une ville inconnue et vous prenez un taxi pour vous rendre chez X c'est le chauffeur de taxi qui, pour répondre à votre requête, prendra le chemin le plus court pour accéder chez X.

Un SGBD se compose de trois couches successives de fonctions, empilées depuis les mémoires secondaires vers les usagers [3,4]:

1. Système de gestion de fichier (SGF) : permet de stocker les données sur les mémoires secondaires.
2. SGBD interne : permet la gestion des liens entre les données stockées dans les fichiers. Qui devient invisible aux programmes d'applications.
3. SGBD externe : assure l'analyse et l'interprétation des requêtes des usagers, et la mise en forme des données échangées avec le monde extérieur.

La figure I.1 illustre ces trois couches de fonctions qui constituent un SGBD.

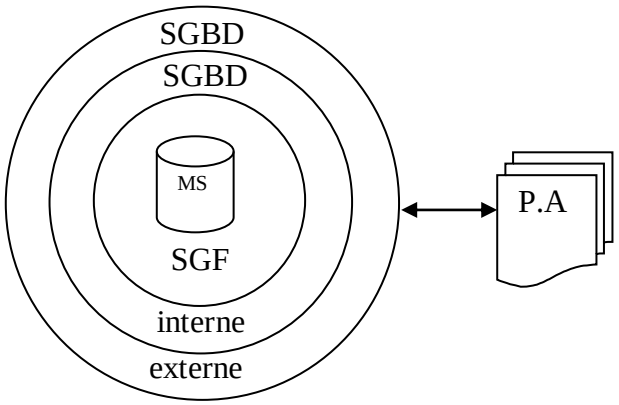


Fig.I.1-les trois couches de fonctions qui constituent un SGBD.
P.A : programme d'application
M.S : mémoires secondaires

I.4. Objectifs d'un SGBD

Les objectifs principaux des systèmes de gestion de base de données sont résumés dans les points suivants [5-8]:

- Indépendance physique
- Indépendance logique
- Manipulation des données par des non-informaticiens
- Efficacité des accès aux données
- Administration cohérente des données
- Non redondance des données
- Cohérence des données
- Partageabilité des données
- Sécurité des données

a. Indépendance physique

C'est un objectif essentiel des SGBD. Il s'agit de permettre de réaliser l'indépendance des structures de stockage aux structures de données du monde réel. Cela consiste à pouvoir définir l'assemblage des données élémentaires entre elles dans le système informatique indépendamment de l'assemblage réalisé dans le monde réel, en tenant compte seulement des critères de performances et de flexibilité d'accès. Ce qui permettrait de 'transporter' plus ou moins facilement des logiciels et des données entre des équipements hétérogènes.

b. Indépendance logique

Il s'agit de permettre à chaque groupe de travail réalisant une application de pouvoir assembler différemment les données pour former des entités et des associations. De plus, chacun doit pouvoir se concentrer sur les éléments constituants son centre d'intérêt, c'est à dire que chacun doit pouvoir ne connaître et ne voir qu'une partie des données.

c. Manipulation des données par des non-informaticiens

Si les objectifs **a** et **b** sont atteints, les utilisateurs non-informaticiens verront les données indépendamment de leur implantation en machine, et peuvent alors les manipuler, les interroger et les mettre à jour, au moyen des langages non procéduraux, c'est à dire en décrivant les données qu'ils souhaitent retrouver sans décrire la manière de les retrouver qui est propre à la machine.

d. Efficacité des accès aux données

Il s'agit d'abord de fournir des langages de manipulation de données très efficaces permettant d'accéder rapidement aux données. D'autre part, pour les non-informaticiens, il doit être possible d'utiliser un langage non procédural dont l'implantation doit être très efficace, notamment au niveau des accès aux disques qui restent le goulot d'étranglement essentiel des SGBD.

e. Administration cohérente des données

L'administration des données est l'ensemble des fonctions privilégiées ayant trait à : la définition des structures de stockage, la création des espaces de travail pour les utilisateurs, l'attribution aux utilisateurs des droits d'accès, …etc. L'extrême importance de ces fonctions, rend obligatoire la mise sous contrôle du SGBD par un ou plusieurs personnes hautement qualifiées (appelées administrateurs du système). L'administration des données est ainsi souvent centralisée. L'objectif est seulement de permettre une administration cohérente et efficace des données.

f. Non redondance des données

Dans les systèmes classiques à fichiers non intégrés, chaque application possède des données propres. Ceci conduit généralement à de nombreuses duplications de données avec, outre la perte de place en mémoire secondaire associée, un gâchis important en moyens humains pour saisir et maintenir à jour plusieurs fois les mêmes données.

Avec une approche 'base de données', les fichiers plus ou moins redondants sont intégrés en un seul fichier partagé par les diverses applications.

g. Cohérence des données

Les données peuvent être soumises à certaines règles. Par exemple, une quantité commandée Q d'un produit P ne doit pas être supérieurs à la quantité S en stock du même produit. Le SGBD doit donc veiller à ce que les applications respectent ces règles lors des modifications des données et doit ainsi assurer la cohérence des données.

h. Partageabilité des données

Cet objectif consiste à permettre aux applications de partager les données de la base dans le temps mais aussi simultanément. Une application doit pouvoir accéder aux données comme si elle était seule à les utiliser, sans attendre mais aussi sans savoir qu'une autre application peut les modifier en même temps.

i. Sécurité des données

Les données doivent être protégées contre les accès non autorisés ou mal intentionnés. Il doit exister des mécanismes pour autoriser, contrôler ou enlever les droits d'accès de n'importe quel usager à un ensemble de données.

Par exemple, un employé pourra connaître les salaires des personnes qu'il dirige, mais pas des autres employés de l'entreprise.

I.5. Concepts et méthodes de base

I.5.1. Notions de base

Type d'objet : ensemble d'objets possédant des caractéristiques similaires et manipulables par des opérations identiques.

Exemples :

- le type d'objet entier muni des opérations {+, -, x, /}
- le type d'objet voiture composé des attributs suivants :
  Voiture (N°V, marque, type, couleur)
  Peut être muni des opérations créer, détruire, modifier, consulter qui sont des opérations standards sur un objet.

Occurrence d'objet : élément d'un ensemble d'objets

Exemple :

- l'entier 10
- la voiture (139 RH 98, Renault, mégane, blanche)

🕮 Toute description de données s'effectue donc au niveau du type à l'aide d'un ensemble d'éléments descriptifs permettant d'exprimer les propriétés d'ensembles d'objets et composant un modèle de description de données.

Modèle de description de données : ensemble de concepts et de règles de composition de ces concepts permettant de décrire des données [6,9,10].

🕮 Un modèle de description de données est souvent représenté par un formalisme graphique. Il est mis en œuvre à l'aide d'un langage de description de données.

Langage de description de données : langage supportant un modèle et permettant de décrire les données d'une base de données d'une manière assimilable par une machine.

🕮 La description d'un ensemble de données particulier, correspondant par exemple à une application, à l'aide d'un langage de description donne naissance à un schéma.

Schéma : description au moyen d'un langage déterminé d'un ensemble de données particulier.

I.5.2. Les niveaux de description des données

Pour assurer l'indépendance, physique et logique, on distingue trois niveaux de descriptions de données [6,7]:

a. Le niveau conceptuel

Correspond à la structure canonique des données qui existent dans l'entreprise, c'est à dire leur structure sémantique inhérente sans souci d'implantation en machine, représentant la vue intégrée de tous les groupes de travail. Par exemple, le schéma conceptuel permettra de définir :

- les types de données élémentaires qui définissent les attributs des objets de l'entreprise (exemple : numéro de véhicule, puissance, couleur. ...)
- les types de données composés qui permettent de regrouper les attributs afin de décrire les entités du monde réel (exemple : personne, voiture…)
- les types de données composés qui permettent de regrouper les attributs afin de décrire les associations du monde réel (exemple : possède……)
- les règles que devront suivre les données au cours de leur vie dans l'entreprise (exemple : tout personne possède une voiture ou plus)

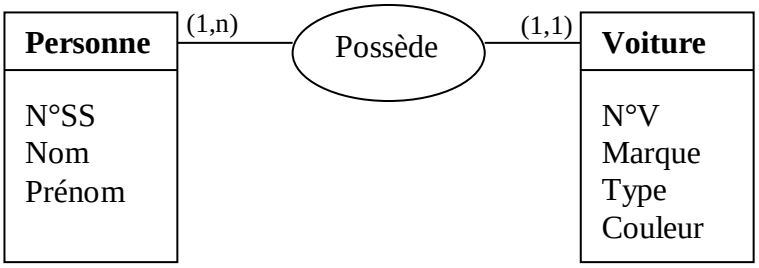


Fig.I.2-exemple de schéma conceptuel

b. Le niveau interne

Correspond à la structure de stockage supportant les données. Il permet donc de décrire les données telles qu'elles sont stockées dans la machine, par exemple :

- les fichiers qui contiennent ces données (nom, organisation, localisation….)
- les articles de ces fichiers (longueur, champs, modes de placement en fichier….)
- les chemins d'accès à ces articles (index, chaînages……)

c. Le niveau externe

Chaque groupe de travail utilisant des données possède une description des données perçues effectuée de la manière dont il les voit dans ses programmes d'application. Alors qu'au niveau conceptuel et interne, les schémas décrivent toute une base de données, au niveau externe ils décrivent simplement la partie des données présentant un intérêt pour un utilisateur ou un groupe d'utilisateurs.

Il est souligné que pour une base de données, il existe un seul schéma interne et un seul schéma conceptuel. Il existe par contre, en générale, plusieurs schémas externes. Ainsi, certains schémas externes peuvent être déduits les uns des autres.

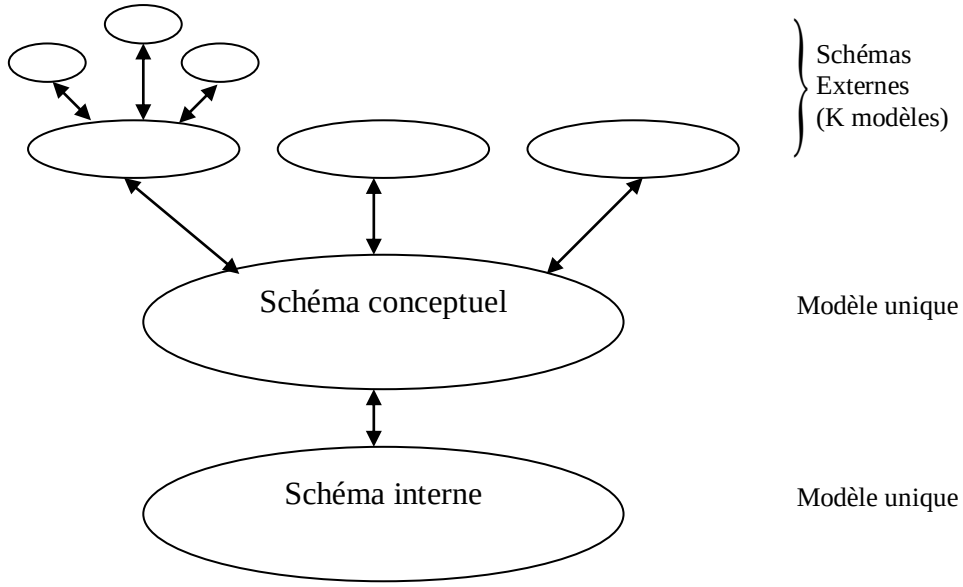


Fig.I.3-différents schémas d'une base de données

Les modèles de données

Nous avons défini un modèle de données comme un ensemble de concepts et de règles de composition de ces concepts permettant de décrire des données. Il existe en fait trois modèles de données principales : le modèle hiérarchique, le modèle réseau et le modèle relationnel. Historiquement les deux premiers modèles ont été conçus dans la période 1965-1970, et ont donné naissance à plusieurs systèmes commercialisés : IMS (Information Management System) utilisant le modèle hiérarchique, IDS (Integrated Data Store) utilisant le modèle réseau … etc.

Nous verrons dans le chapitre suivant que le modèle relationnel a remplacé progressivement ces deux modèles cités.

II.1. Le modèle réseau

Le modèle réseau a été proposé par le groupe DBTG du comité CODASYL Il a été amélioré à plusieurs reprises grâce à des études qui ont abouti surtout à sa normalisation.
Les éléments de base de ce modèle sont les notions d'ensembles d'entités (enregistrement logique) et d'associations entre ensembles d'entités (liens) [8].

Le premier élément fondamental dans le modèle réseau est la notion d'enregistrement logique n'est pas spécifique aux SGBD, c'est celle que l'on rencontre dans la plupart des systèmes de gestion de fichiers et des langages de programmation. Elle permet de regrouper des données élémentaires en enregistrements logiques.

Exemple :
```
Etudiant = enregistrement
 Numéro_etudiant : entier, clé
 Nom             : caractère(10)
 Adresse         : caractère(40)
 Fin
```

Le deuxième élément fondamental dans le modèle réseau est le lien. Un lien peut se définir comme la représentation d'une association, l'association est une perception abstraite de la réalité alors que le lien va être sa matérialisation [8].

Un lien L est défini entre deux types d'enregistrements R et S dans une direction donnée. Il offre un mécanisme d'accès qui, à partir d'un enregistrement r de type R, permet d'accéder aux enregistrements s_1, s_2, …s_n de type S qui lui sont reliés. Conventionnellement, un lien est représenté par un diagramme appelé *diagramme de bachman*, du nom de l'inventeur, où sont mentionnés les types d'enregistrement R et S et le lien L qui va de R vers S, comme montre la figure suivante.

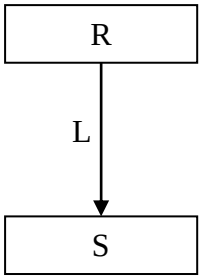


Fig.II.1-Diagramme de bachman

Par exemple, soit le diagramme suivant (fig.II.2) qui associé les entités ‘étudiant’ et ‘formation’ par le lien ‘est inscrit à’

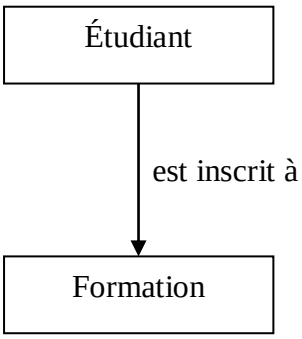


Fig.II.2- exemple d’entités et de lien

Si à chaque occurrence de l’entité R, il existe une et une seule occurrence de l’entité S, le lien est dit de type : 1-1, par exemple : un étudiant ne peut s’inscrire qu’à une seule formation.

Si on peut associer à chaque occurrence de l’entité R plusieurs occurrences de l’entité S, le lien est dit de type : 1-n, par exemple : un père peut avoir plusieurs fils.

Si à plusieurs occurrences de l’entité R, on peut associer une seule occurrence de l’entité S, le lien est dit de type : n-1, par exemple : plusieurs enfants sont les descendants d’un seul père.

Si à une occurrence de l’entité R, on peut associer n occurrences de l’entité S, et inversement, à une occurrence de l’entité S, on peut associer m occurrences de l’entité R, le lien est dit de type : m-n. par exemple, une formation peut faire appel à n matières, et inversement, une matière peut faire partie de m formations différentes. Nous verrons dans le paragraphe suivant que ce type de lien pose un problème d’implémentation. En effet, plusieurs techniques ont été proposées pour le représenter, ce qui a conduit à des variantes du modèle réseau : modèle réseau CODASYL, modèle réseau SOCRATE….etc.

II.2. Le modèle réseau CODASYL

Le lien appelé ‘SET’ dans la terminologie CODASYL, ne peut être utilisé que lorsque l’association qui existe entre les enregistrements de type R et S fait correspondre à un enregistrement r les enregistrements s_1, s_2.... ,s_n alors qu’inversement à chaque élément s_I ne correspond au plus qu’un élément r. Dans ce cas, on pourra construire un lien directionnel de R vers S [11-13].

Cela veut dire, qu’un lien CODASYL pourra être construit si l’association est de type 1-1, 1-n ou n-1. Dans le cas d’une association de type m-n, il faudrait introduire un enregistrement de nouveau type (voir ci-dessous) ; avant cela, expliquons comment se fait le mécanisme d’accès.

Le mécanisme d'accès qui permet de passer de l'enregistrement r aux enregistrements s_1, s_2, ….,s_n est celui d'une liste circulaire (voir fig.II.3) où la tête de liste est constituée par l'enregistrement r et les éléments de la liste sont constitués par les enregistrements s_1, s_2, ….,s_n

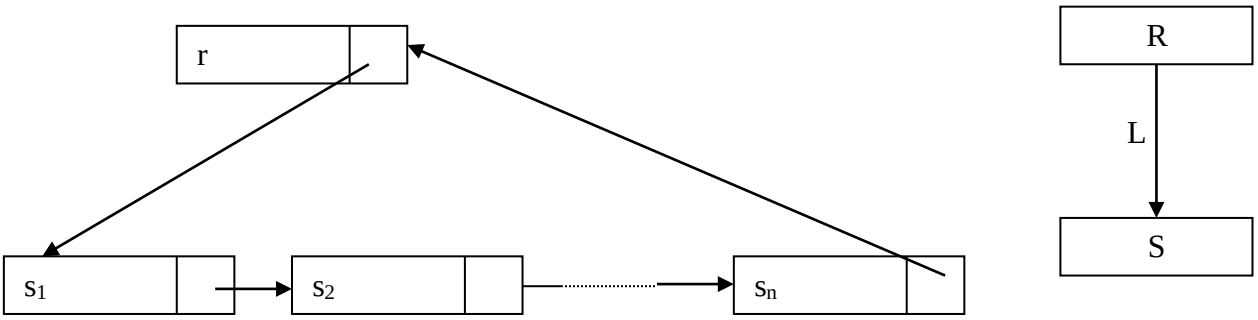


Fig.II.3-représentation d'un lien CODASYL à l'aide d'une liste circulaire

Cette représentation est possible car chaque enregistrement de type S n'étant associé qu'à un seul enregistrement de type R, et ne peut apparaître que dans une seule liste circulaire. En effet, si un enregistrement de type S était parcouru par plusieurs listes, il y aurait un nombre variable de pointeurs et c'est la raison pour laquelle une association de type m-n ne peut être représentée par un lien CODASYL sans une transformation préalable.

Dans le cas d'une association de type m-n entre deux types d'enregistrements R et S, la solution consiste à se ramener à deux associations de type 1-n en construisant un enregistrement de type noté RxS (figII.4) dont les éléments sont tous les couples de la forme (r_I,s_j) si r_I est de type R, s_j de type S et r_I et s_j étant reliés entre eux dans l'association. Si on examine maintenant les associations qui existent respectivement entre RxS et R et S, on remarque qu'elles sont toutes deux de type 1-n

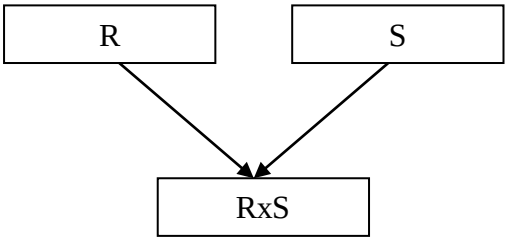


Fig.II.4-représentation de l'association m-n en formalisme CODASYL

II.2.1. Schémas réseaux

La notion de lien CODASYL telle qu'elle vient d'être définie va permettre de construire des schémas réseaux (fig.II.5) dont la structure sera spécifique de chaque application, mais qui d'une façon générale auront les propriétés suivantes [14-16]:

a- d'un enregistrement de type R peuvent partir autant de liens différents que l'on veut
b- à un enregistrement de type S peuvent arriver autant de liens que l'on veut
c- entre deux enregistrements de type R et S, il peut y avoir plusieurs liens différents allant de R vers S, et inversement
d- pour un enregistrement de type R, un lien partant de R ne peut boucler sur R. Cette limitation a en fait été levée dans certains systèmes commercialisés.

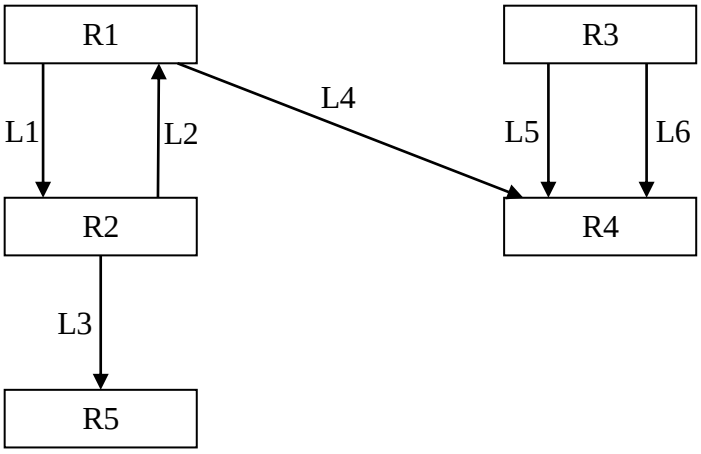


Fig.II.5-exemple de diagramme réseau CODASYL

A titre d'exemple, nous allons présenter le diagramme CODASYL d'une base de données comme dans la figure suivante (fig.II.6) composée des enregistrements suivants :
-Produit composé des données élémentaires suivantes : Numéro-Prod et Nom-Prod
-Fournisseur composés des données : Numéro-Fourn, Nom-Fourn et Prénom-Fourn
-Acheter décrivant pour chaque produit, la quantité acheté par un fournisseur
-Producteur définissant pour chaque produit, le nom du producteur
-Commandes spécifiant les commandes de produit passées par les fournisseurs aux producteurs

Les liens existants entre ces différents enregistrements sont :
-Production qui associe un producteur aux produits
-Vente qui associe un produit aux commandes correspondantes
-Achat qui associe un fournisseur à ses commandes
-Produire qui associe un fournisseur à une quantité de produit
-Consommation qui associe un produit à toutes ses quantités produit par les différents fournisseurs.

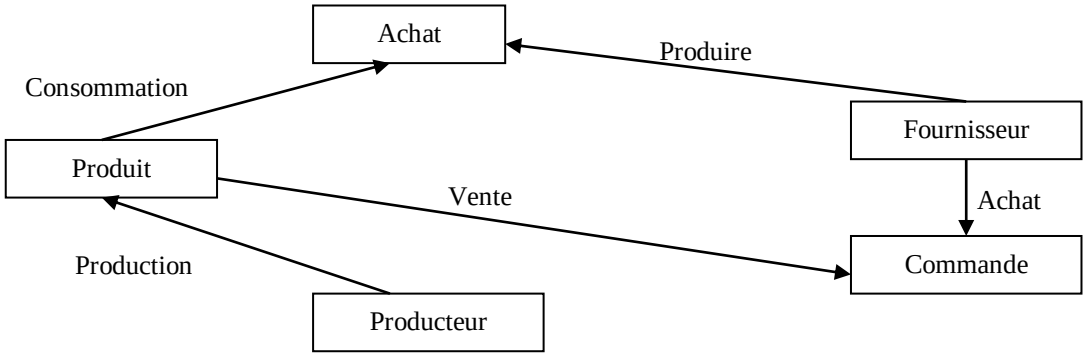


Fig.II.6-Diagramme d'une BDD réseau

II.2.2. Les opérations de manipulation dans un modèle réseau

Il y a deux grandes catégories d'opérations [14,15]: celles qui permettent de modifier le contenu de la base de données et celles qui permettent de sélectionner de l'information

Les opérations qui permettent de modifier le contenu de la BDD sont les opérations de création, de suppression et de mise à jour

Pour le modèle CODASYL, considérons un enregistrement r de type R et un enregistrement s de type S, et supposons que nous voulions ajouter le couple (r,s) comme appartenant au lien L qui va de R à S [11-13].

Si les enregistrements r et s n'existent pas dans la BDD, la première étape consiste à insérer ces deux enregistrements dans les fichiers relatifs aux enregistrements de type R et S, en respectant l'organisation des données relatives à ces deux fichiers.

La deuxième étape consistera à construire une liste circulaire ayant pour point de départ l'enregistrement r.

Pour ce qui est de la recherche d'information, le programmeur doit utiliser la connaissance qu'il a des chemins d'accès de la base de données.

3. Le modèle Hiérarchique

Les éléments de base du modèle hiérarchique sont les enregistrements logiques qui sont reliés entre eux pour constituer une arborescence. Une arborescence est un cas particulier d'un graphe réseau dans lequel un nœud, peut recevoir au maximum une flèche, et il peut appartenir à plusieurs flèches, comme le montre la figureII.7.

Sur l'arborescence, il existe un nœud particulier : la racine, c'est un nœud qui ne reçoit aucune flèche. On appelle fils d'un nœud l'ensemble des nœuds qui se trouvent à une distance 1 de ce nœud. On appelle père d'un nœud, la notion inverse de celle de fils. Dans une arborescence, tout nœud n'a qu'un seul père et peut avoir plusieurs fils.

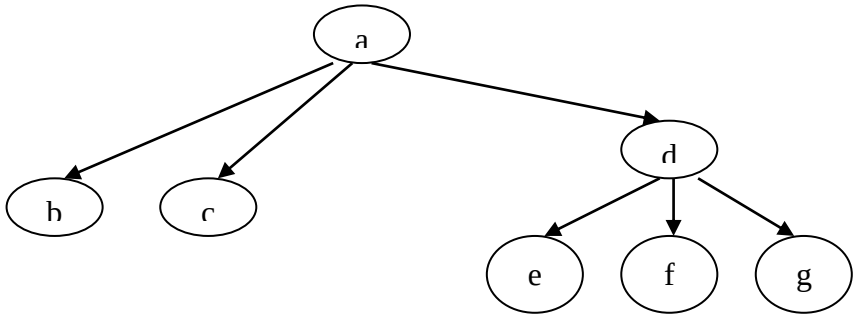


Fig.II.7-une arborescence

4. Conclusion

Dans ce chapitre, nous avons introduit les deux principaux modèles de données, réseau et hiérarchique, qui ont existé bien avant le modèle relationnel. Compte tenu de l'accroissement des possibilités des ordinateurs, constaté à partir des années70, ces modèles sont actuellement critiqués. Ils peuvent rester très compétitifs au niveau interne. Les critiques proviennent surtout de la complexité des langages de manipulations associés à ces modèles, basé sur la *'navigation'*, donc peu accessibles aux utilisateurs néophytes.

Finalement, la question se pose de savoir si le modèle relationnel est un bon candidat pour constituer un modèle conceptuel. D'après Codd, l'inventeur du modèle relationnel, un bon modèle doit avoir quatre caractéristiques :

1- Permettre une représentation sous forme de tableaux des données.
2- Permettre une interprétation ensembliste afin d'autoriser des opérations similaires à ceux de l'algèbre relationnelle.
3- Permettre une interprétation des objets en termes de formules de la logique afin de faciliter les déductions de connaissances.
4- Supporter une représentation graphique afin de pouvoir visualiser les objets et leurs associations pour concevoir la base.

Ajoutons que la simplicité est une caractéristique essentielle du modèle relationnel qui, dans sa version entité-association, possède les quatre caractéristiques précédentes.

Modèle relationnel

III.1. Introduction

Nous avons déjà mentionné qu'il existe trois grandes catégories de modèles de données : le modèle réseau, le modèle hiérarchique et le modèle relationnel. Les deux premiers sont de plus en plus abandonnés actuellement.

Le modèle relationnel a été inventé par CODD à IBM-San Jose en 1970. Il est basé sur la théorie de la relation et de la normalisation des relations [17].

III.2. Les concepts de base du modèle

Le concept de relation découle directement de la théorie des ensembles. Nous allons en rappeler la définition qui nécessite tout d'abord l'introduction de la notion de domaine [17-19].

Notion de Domaine : *C'est l'ensemble de valeurs caractérisées par un nom.*

A titre d'exemple, on peut considérer le domaine des entiers, celui des réels, celui des valeurs logiques {vrai, faux}, celui des jours de la semaine {samedi, dimanche, lundi, mardi, mercredi, jeudi, vendredi}, celui des couleurs du drapeau algérien {rouge, vert, blanc}…etc.

Rappelons que le produit cartésien d'un ensemble de domaines D1, D2, ….,Dn, noté D1xD2x….xDn, est l'ensemble des tuples $<v_1, v_2, \ldots..,v_n>$ tels que $v_i \in Di$.
C'est à dire c'est la totalité des combinaisons possibles que l'on peut tirer de ces domaines.
Par exemple, le produit cartésien des domaines pays={Algérie, France, Espagne} et capitale={Alger, Paris, Madrid} est composé des tuples suivants comme montre le tableau suivant:

Algérie	Alger
Algérie	Paris
Algérie	Madrid
France	Alger
France	Paris
France	Madrid
Espagne	Alger
Espagne	Paris
Espagne	Madrid

Table.III.1-Produit cartésien d'un ensemble de domaines

Notion de relation : *Sous-ensemble du produit cartésien d'une liste de domaines*
Une relation est généralement caractérisée par un nom.

Par exemple, à partir des domaines :
D1={Algérie, France, Espagne} et D2={Alger, Paris, Madrid}
On peut composer une relation qui nous intéresse, comme montre le tableau suivant, c'est à dire celui qui a un sens par rapport à la réalité :

Pays_capital	**D1**	**D2**
	Algérie	Alger
	France	Paris
	Espagne	Madrid

Table.III.2- Sous-ensemble du produit cartésien

- Plus simplement, une relation peut être vue comme un tableau à deux dimensions dont les colonnes correspondent aux domaines et les lignes contiennent les tuples. On parle parfois de ***table relationnelle***.

Notion de tuple (n-uplet) : *C'est une ligne d'une relation*

Par exemple, (Algérie, Alger)
- Afin de rendre l'ordre des colonnes sans importance tout en permettant plusieurs colonnes de même domaine, on associe un nom à chaque colonne. On parle parfois *d'enregistrement.*

Notion d'attribut : *C'est une colonne d'une relation caractérisée par un nom*
- On parle parfois de *constituant* ou *composant*

Notion de schéma de relation : *C'est le nom d'une relation suivi de la liste des attributs qui la constituent.*

Par exemple, une relation décrivant des voitures et comportant les attributs : N°V, marque, type, puissance, couleur, aura pour schéma :
VOITURE (N°V, marque, type, puissance, couleur)

- A toute relation, il est possible d'associer un schéma de relation qui représente l'intention de la relation, alors que le tableau avec les tuples représente une extension.

Nous pouvons maintenant introduire la notion de base de données relationnelle

Notion de base de données relationnelle : *C'est une base de données dont le schéma est un ensemble de schémas de relations et dont les occurrences sont des tuples de ces relations.*

III.3. Introduction a la conception de schémas relationnels

III.3.1 Perception du monde réel

Avant de construire le modèle relationnel, le monde réel peut être modélisé au niveau conceptuel à l'aide du modèle Entité/Association (E/A) qu'est une représentation graphique des données.
Les concepts utilisés dans le modèle E/A sont-les :
Entité : qui représente un objet ayant une existence visible
Association : est une relation qui s'établit entre deux entités ou plus.
Cardinalité : c'est le nombre minimal et maximal d'occurrences d'une association
Propriété : est une information décrivant une caractéristique d'une entité ou d'une association.

A titre d'exemple, soit des données modélisant des entités 'personne' et 'voiture', et le type d'association 'possede' qui traduit le fait qu'une personne soit propriétaire d'une ou plusieurs voitures. Une personne est caractérisée par un numéro de sécurité sociale (N°SS), un nom et un prénom alors qu'une voiture est caractérisée par les propriétés déjà vues N°V, marque, type, puissance, et couleur. Chaque personne est identifiée par une occurrence du N°SS alors que chaque voiture est identifiée par son N°V. A chaque occurrence d'association correspond par exemple une date d'achat (date) et un prix d'achat (prix). La figure.III.1illustre la partie du monde réel modélisé par le modèle E/A en représentant une entité avec leurs propriétés par un rectangle, une association avec leurs propriétés par un hexagone.

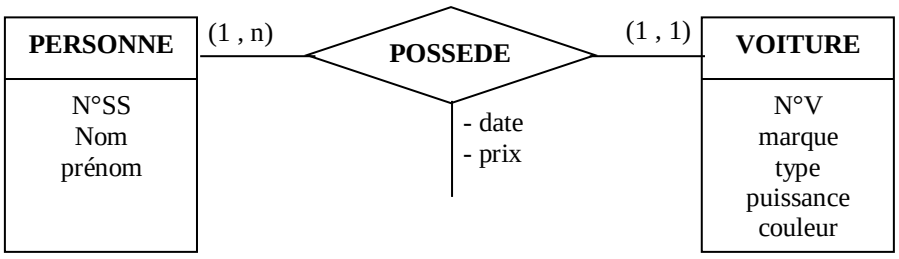


Fig.III.1-Représentation du monde réel par des entités et des associations

III.3.2. Les règles de passage du niveau conceptuel au modèle relationnel

Le modèle relationnel se prête bien à la représentation des entités et des associations [18-21] :
- Une propriété devient un attribut
- Une entité est représentée par une relation (table) dont le schéma est le nom de l'entité suivi de la liste des attributs de l'entité et son identifiant devient sa clé primaire.
- Une association devient une relation et on distingue deux cas selon le type de l'association :
  a) Relation binaire de type père fils : ce sont donc des relations binaires (2 dimensions) pour les quelles les cardinalités de l'entité père sont 0-n / 1-n et les cardinalités de l'entité fils sont 0-1 / 1-1
     - les propriétés de l'association deviennent des attributs de la relation fils
     - la relation disparaît
     - l'identifiant de l'entité père devient la clé secondaire (étrangère) de la relation fils

A titre d'exemple :

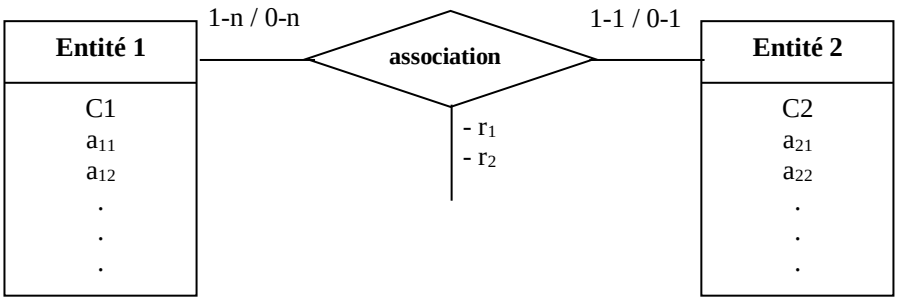


Fig.III.2-Représentation d'une relation binaire de type père fils

Résultat : relation entité1(C1, a_{11}, a_{12}, …..)
relation entité2(C2, C1, a_{21}, a_{22}, …..r_1, r_2)

b) autre relation : Pour les autres relations, la clé d'une telle relation est composée de l'ensemble des identifiants des entités qui participent à l'association du formalisme individuel.

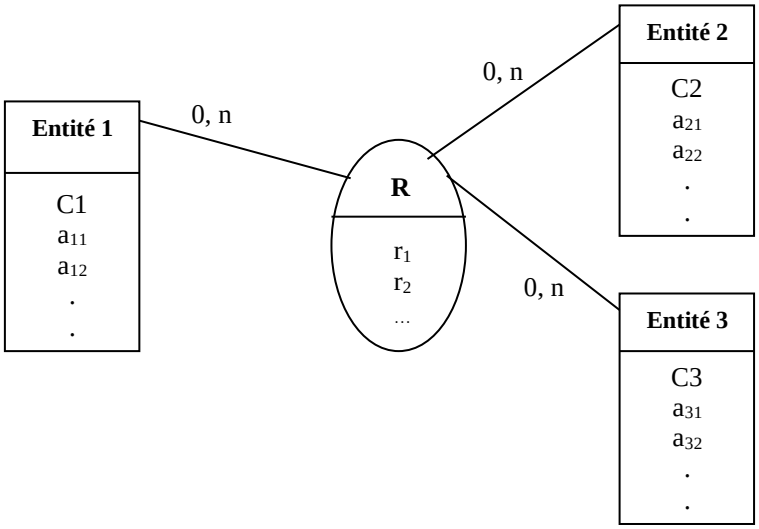


Fig.III.3-Représentation d'une relation n dimensions

Résultat : relation entité1(C1, a_{11}, a_{12}, …..)
relation entité2(C2, a_{21}, a_{22}, …..)
relation entité3(C3, a_{31}, a_{32}, …..)
relation R(C1, C2, C3, r_1, r_2, …)

III.3.3. Problèmes soulevés par une mauvaise perception du monde réel

Une mauvaise conception des entités et des associations représentant le monde réel modélisé conduit à des relations problématiques. Imaginons par exemple que l'on concevra une entité unique propriétaire contenant tous les attributs des trois relations personne, voiture et possede. Le tableau suivant représente une extension possible de cette relation.

N°V	Marque	Type	Puissance	Couleur	N°SS	Nom	Prénom	Date	Prix
342.185	Peugeot	406	9	Verte	100	Omari	M^{ed}	12-12-90	400000DA
155.188	Renault	R12	6	Grise	150	Hamidi	Rafik	25-05-89	300000DA
226.186	Renault	R19	9	Rouge	100	Omari	M^{ed}	12-12-90	600000DA
145.187	Renault	R21	9	Jaune	150	Hamidi	Rafik	25-05-89	500000DA
697.188	Peugeot	305	9	Verte	200	Saleh	Ahmed	10-10-89	450000DA

Table.III.3- Représentation du monde réel par une mauvaise conception

Cette relation souffre de plusieurs types d'anomalies :
- Tout d'abord, des données sont redondantes ; par exemple, Omari et Hamidi apparaissent deux fois ; plus généralement, une personne apparaît autant de fois qu'elle possède de voitures
- Ces redondances conduisent à des risques d'incohérence lors d'une mise à jour ; par exemple, si l'on s'aperçoive que le prénom de Hamidi n'est pas Rafik mais Mostefa, il faudra veiller à mettre à jour tous les tuples contenant Hamidi
- Il est nécessaire d'autoriser la présence de valeurs nulles dans une telle relation afin de pouvoir conserver les personnes qui n'ont plus de voitures.

En résumé, une relation qui ne représente pas de vraies entités ou associations semble donc souffrir de présence de données redondantes et d'incohérences potentielles et nécessite le codage de valeurs nulles. Il y a tout intérêt à éliminer ces anomalies afin de faciliter la manipulation des relations.

III.3.4. L'approche par décomposition

L'approche par décomposition pour concevoir des schémas relationnels tend à partir d'une relation composée de tous les attributs, appelée relation universelle, à décomposer cette relation en sous-relations qui ne souffriraient pas des anomalies précédemment signalées. Il doit être réalisé par un algorithme à partir d'une bonne compréhension des propriétés sémantiques des données. Cette approche (fig.III.4) utilise un algorithme de décomposition [21-23].

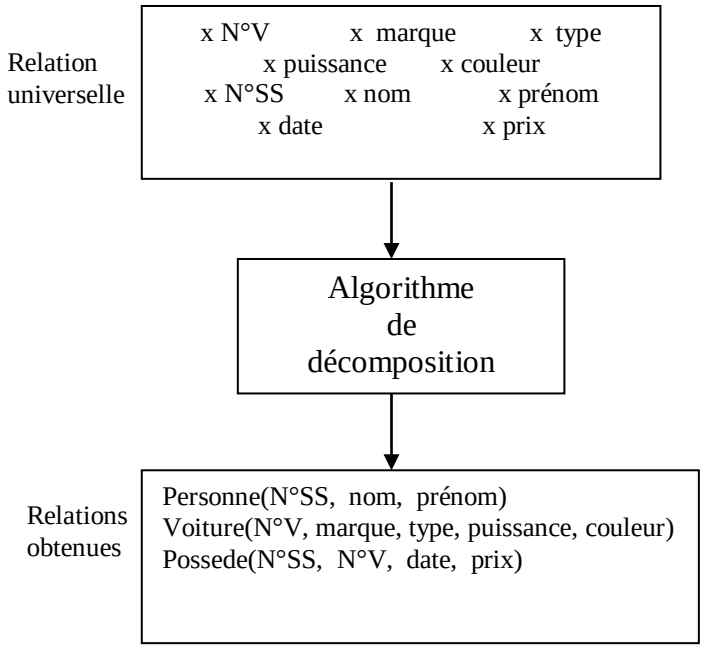


Fig.III.4-Décomposition d'une relation universelle

La compréhension de la théorie de décomposition des relations nécessite la connaissance de deux opérations de base : la projection et la jointure.

La projection est l'opération qui consiste à supprimer des attributs d'une relation et à éliminer les tuples en double qui risquent d'apparaître dans la nouvelle relation obtenue. La projection de la relation R de schéma $R(A_1, A_2, \ldots..,A_n)$ sur les attributs $A_1, A_2, \ldots..,A_p$ est une relation R' de schéma $R'(A_1, A_2, \ldots..,A_p)$ obtenue par élimination des attributs de R n'appartenant pas à R' et suppression des tuples en doubles.
Une telle projection R' sera notée : $R' = \Pi\ (A_1, A_2, \ldots..,A_p)[R]$
Par exemple, la projection de la relation propriétaire sur les attributs nom et prénom donne la relation suivante :

Π (nom, prénom)[propriétaire]	Nom	Prénom
	Omari	M^{ed}
	Hamidi	Rafik
	Saleh	Ahmed

Table.III.4- Projection d'une relation

La jointure naturelle ou simplement jointure, est l'opération inverse de la projection. La jointure de deux relations R et S, de schéma respectif R(A_1, A_2, ….,A_n) et S(B_1, B_2,…,B_p) est une relation T, notée R ⋈ S ayant pour attribut l'union des attributs de R et S, soit
{ A_1, A_2, ….,A_n} ∪ { B_1, B_2,…,B_p}
et pour tuples tous ceux obtenus par concaténation des tuples de R et S ayant même valeurs pour les attributs de même nom. Ainsi, on a :
Π A_1, A_2, ….,A_n(T)=R
Π B_1, B_2,…,B_p(T)=S
L'exemple suivant fait la jointure de deux relations R et S :

R	marque	couleur
	Renault	Rouge
	Peugeot	Verte
	Citroen	Bleue
	Renault	Verte

S	couleur	puissance
	Rouge	6
	Verte	9
	Bleue	2
	Bleue	5
	verte	6

R ⋈ S	marque	couleur	puissance
	Renault	Rouge	6
	Peugeot	Verte	9
	Peugeot	Verte	6
	Citroen	Bleue	2
	Citroen	Bleue	5
	Renault	Verte	9
	Renault	verte	6

Table.III.5- Jointure deux relations R et S

Il est maintenant possible de définir précisément la notion de décomposition :

Notion de décomposition : *Un schéma de relation R(A_1, A_2, ….A_n) est convenablement décomposée en une collection de relations R1, R2,…,Rp, obtenues par des projections de R ,si la jointure R1 ⋈ R2 ⋈…… ⋈Rp ait même schéma que R.*

Par suite, lors d'une décomposition, le schéma de relation R(A_1, A_2, ….A_n) est remplacé par une collection de schémas dont l'union des attributs est (A_1, A_2, ….A_n). Il est important que la décomposition se fasse sans perte d'informations.

Notion de décomposition sans perte : *Décomposition d'une relation R en R1, R2,…,Rp, tel que pour toute extension de R, on ait : R=R1 ⋈R2 ⋈…. ⋈Rp*

🕮 C'est à dire, après la décomposition de relation R en une collection de relations R1, R2,…,Rp, si la jointure R1 ⋈ R2 ⋈ …. ⋈ Rp donne la même extension que R on dit que la décomposition sans perte, sinon, si donne seulement les mêmes attributs en dits que c'est une décomposition.

Le problème de la conception des bases de données relationnelles peut donc être vu comme celui de décomposer la relation universelle composée de tous les attributs en sous-relations ne souffrant pas des anomalies (vues au précédent) et de sorte à obtenir une décomposition sans perte. Nous allons ci-dessous étudier les principales méthodes proposées pour effectuer une telle décomposition qui doit permettre de déterminer des entités et associations canoniques du monde réel, donc en fait de construire un schéma conceptuel.

III.4. Les dépendances fonctionnelles(DF)

III.4.1. Notion de dépendance fonctionnelle

La notion de dépendance fonctionnelle fut introduite par CODD afin de caractériser des relations qui peuvent être décomposées sans perte d'informations [22-24].

Dans une relation, un attribut B dépend fonctionnellement d'un attribut A, si pour une valeur de A, il ne peut exister, qu'une valeur de B.
B dépend fonctionnellement de A, si, connaissant A, je connais une valeur de B.

Définition :
Soit $R(A_1, A_2, ...A_n)$ un schéma de relation et X et Y des sous-ensembles de $\{ A_1, A_2, ...A_n\}$. On dit que X détermine Y ou Y dépend fonctionnellement de X, si étant donné une valeur de X, il lui correspond une valeur unique de Y(et ceci quelque soit l'instant considéré).

A titre d'exemple, dans la relation voiture, on a les dépendances fonctionnelles :
N°V $\rightarrow$ couleur
type $\rightarrow$ marque
type$\rightarrow$ puissance
(type,marque) $\rightarrow$ puissance

Il est très important de remarquer qu'une dépendance fonctionnelle (DF) doit être vraie sur toutes les valeurs possibles et non sur les valeurs actuelles. Autrement dit, il est impossible de déduire les DF d'une réalisation particulière d'une relation. La seule manière de déterminer une DF est de regarder soigneusement ce que signifient les attributs car ce sont les déductions sur le monde réel qui lient les valeurs possibles des attributs entre elles. Les DF doivent être décelées par le concepteur au niveau du schéma conceptuel.

III.4.2. Propriétés des dépendances fonctionnelles

Les DF obéissent à plusieurs règles [21,24] :

1. **Réflexivité :** $Y \subseteq X \quad \Rightarrow \quad X \rightarrow Y$; cette règle stipule que tout ensemble d'attributs détermine lui-même ou une partie de lui-même.

2. **Augmentation :** $X \rightarrow Y \ \Rightarrow \ XZ \rightarrow YZ$; cette règle signifie que si X détermine Y, les deux ensembles d'attributs peuvent être enrichis par un troisième.

3. **Transitivité :** $X \rightarrow Y$ et $Y \rightarrow Z \ \Rightarrow \ X \rightarrow Z$

Les trois règles précédentes composent les axiomes des DF. Plusieurs autres règles se déduisent des axiomes :

4. **Union :** $X \rightarrow Y$ et $X \rightarrow Z \Rightarrow X \rightarrow YZ$

5. **Pseudo-transitivité :** $X \rightarrow Y$ et $WY \rightarrow Z \Rightarrow WX \rightarrow Z$

6. **Décomposition :** $X \rightarrow Y$ et $Z \subseteq Y \Rightarrow X \rightarrow Z$

A partir de ces règles, il est possible d'introduire la notion de dépendance fonctionnelle élémentaire :

Notion de Dépendance Fonctionnelle Elémentaire
C'est une DF de la forme $X \rightarrow A$, où A est un attribut unique non inclus dans X ($A \not\subset X$) et où il n'existe pas $X' \subset X$ tel que $X' \rightarrow A$

La seule règle d'inférence qui s'applique aux dépendances fonctionnelles élémentaires est la transitivité.

III.4.3 Graphe des dépendances fonctionnelles

Pour mieux identifier les dépendances fonctionnelles élémentaires, on a recours à une représentation graphique.

Par exemple, considérons les dépendances fonctionnelles entre les attributs de la relation VOITURE :

F={N°V→type, type→marque, type→puissance, N°V→couleur}

La figure suivante représente le graphe associé aux dépendances fonctionnelles de F

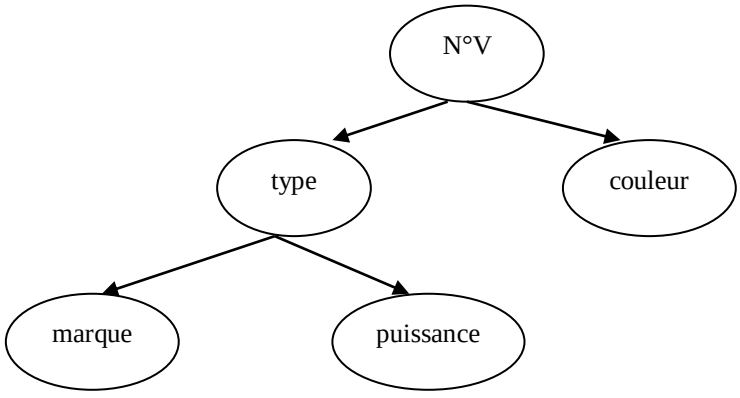


Fig.III.5- Représentation un graphe associé aux dépendances fonctionnelles

Autre exemple : soit la relation suivante code_postal(code, ville, rue), comportant les DF suivantes :
(ville, rue) $\rightarrow$code
code $\rightarrow$ville
Le graphe de ces dépendances fonctionnelles est représenté par la figure suivante :

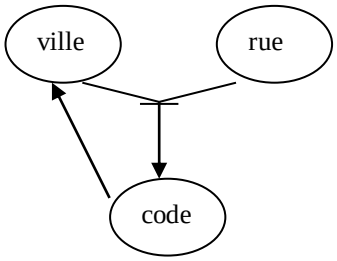


Fig.III.6- Représentation un graphe associé aux dépendances fonctionnelles particulier

III.4.4. Fermeture transitive et couverture minimale

A partir d'un ensemble de DF élémentaires, on peut composer par transitivité d'autres DF élémentaires. On aboutit ainsi à la notion de fermeture transitive d'un ensemble F de DF élémentaires [23,24].

Notion de fermeture transitive
C'est l'ensemble des DF élémentaires considérées enrichis de toutes les DF élémentaires déduites par transitivité.

Par exemple, à partir de l'ensemble de DF :
F={N°V$\rightarrow$type, type$\rightarrow$marque, type$\rightarrow$puissance, N°V$\rightarrow$couleur}
On déduit la fermeture transitive :
F^{+}=F$\cup${N°V$\rightarrow$marque, N°V$\rightarrow$puissance}

Le graphe correspondant à la fermeture transitive de F est :

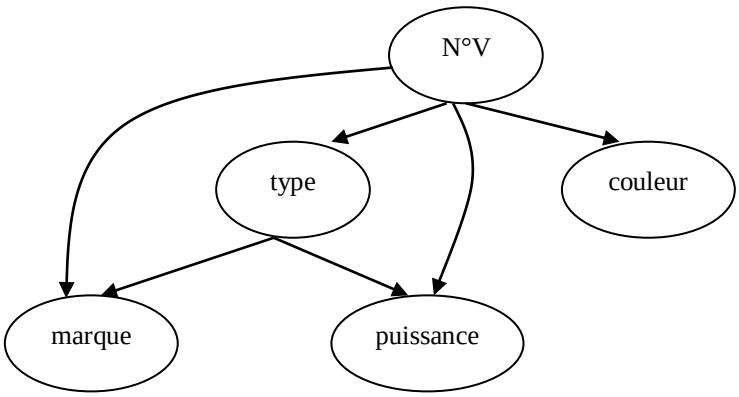


Fig.III.7- Représentation un graphe associé à la fermeture transitive

A partir de la notion de fermeture transitive, il est possible de définir l'équivalence de deux ensembles de DF élémentaires : deux ensembles sont équivalents s'ils ont la même fermeture transitive.

D'autre part, il est intéressant de déterminer un sous-ensemble minimum de DF permettant de générer toutes les autres.

Notion de couverture minimale
C'est un ensemble de DF élémentaires associé à un ensemble d'attributs vérifiant les propriétés suivantes.
1-aucune DF n'est redondante.
2-toute DF élémentaire des attributs est dans la fermeture transitive.

Par exemple : F={N°V→type, type→marque, type→puissance, N°V→couleur}
est une couverture minimale pour l'ensemble des DF de la base de données voiture.

La couverture minimale est essentielle pour décomposer les relations sans perte d'informations.

III.5. Notion de clé et trois premières formes normales

III.5.1 Notion de clé d'une relation

C'est un sous-ensemble X des attributs d'une relation $R(A_1, A_2, ...,A_n)$ telque
1-$X \rightarrow A_1, A_2, ...,A_n$
2-Il n'existe pas de sous-ensemble $Y \subset X$ telque $Y \rightarrow A_1, A_2, ...,A_n$

En clair, une clé est un ensemble minimum d'attributs qui détermine tous les autres. Par exemple, N°V est une clé de la relation voiture alors que (N°V, type) n'est pas une clé.
Il peut y avoir plusieurs clés pour une relation, on les appelle clés candidates, parmi elles, l'une sera choisie pour être la *clé primaire,* les autres clés candidates sont des clés secondaires [21-24].

III.5.2. Définition des trois premières formes normales

Les trois premières formes normales ont pour objectif de supprimer toutes les redondances et les anomalies dans la base de données et de permettre la décomposition de relations en plusieurs relations plus petites sans perdre d'informations, à partir de la notion de dépendance fonctionnelle.

📖 Notons qu'en arrivant au stade du MCD validé dans la méthode de merise, ce dernier est en 2ème forme normale d'après les règles de la normalisation au niveau du MCD brut. Mais, il reste la 3ème forme normale malgré qu'il en existe cinq formes normales, nous ne intéressons pas dans ce qui suit aux 4ème forme normale et 5ème forme normale.

💣 Notons aussi que si une table relationnelle doit être en 3ème forme normale, il faudrait qu'elle soit en 1ère forme normale et 2ème forme normale.

La première forme normale permet simplement d'obtenir des valeurs d'attributs non décomposables.

Notion de première forme normale (1FN)

Une relation est en première forme normale, si tout attribut contient une valeur atomique.

Cette forme normale consiste simplement à éviter les domaines composés de plusieurs valeurs. Par exemple, la relation personne(nom, prénom) sera ainsi décomposée en personne1(nom, prenom1) et personne2(nom, prenom2).

La deuxième forme normale permet d'assurer l'élimination de certaines redondances en garantissant qu'aucun attribut n'est déterminé seulement par une partie de la clé.

Notion de deuxième forme normale (2FN)

Une relation est en deuxième forme normale si et seulement si :
1-elle est en première forme normale
2-chaque attribut non identifiant, dépend fonctionnellement de la clé dans sa totalité

Par exemple : soit la relation etudiant
ETUDIANT(nom, adresse, matière, note)
Nous remarquons qu'il existe une anomalie dans la relation etudiant, en effet la 2ème FN n'est pas vérifiée :
L'attribut adresse dépend d'une partie de la clé 'nom' et non pas de toute la clé
La solution est d'éclater la relation etudiant en deux tables relationnelles en 2FN :
ETUDIANT(nom, adresse)
EXAMEN(nom, matière, note)

La troisième forme normale permet d'assurer l'élimination des redondances dues aux dépendances transitives.

Notion de troisième forme normale (3FN)

Une relation est en troisième forme normale, si et seulement si :
1-elle est en deuxième forme normale
2-tout attributs n'appartenant pas à une clé ne dépendent pas d'un attribut non clé.

Cela signifie qu'il n y a pas de dépendance fonctionnelle transitive dans la relation.
A titre d'exemple, la relation :
VOITURE(N°V, marque, type, puissance, couleur)

n'est pas en troisième forme normale. En effet, l'attribut non clé type détermine marque et aussi puissance. Cette relation peut être décomposée en deux relations :
VOITURE(N°V, type, couleur)
MODELE(type, marque, puissance)

III.5.3. Propriétés d'une décomposition en troisième forme normale

Les DF sont des règles indépendantes du temps que doivent vérifier les valeurs des attributs. Il est nécessaire qu'une décomposition préserve ces règles.

Notion de décomposition préservant les DF
Décomposition {R1, R2, ...,Rn} d'une relation R tel que la fermeture transitive des DF de R est la même que celle de l'union des DF de {R1, R2, ...,Rn}.

D'où l'importance de la troisième forme normale. En effet, toute relation a au moins une décomposition en troisième forme normale telle que :
1-la décomposition préserve les DF
2-la décomposition est sans perte
cette décomposition peut ne pas être unique

III.5.4. Algorithme de décomposition en troisième forme normale

Il existe donc au moins une décomposition en troisième forme normale préservant les DF et sans perte. Une telle décomposition peut être produite par un algorithme ayant pour entrées l'ensemble des attributs ainsi que les DF (voir figure suivante).

Fig.III.8- Algorithme de décomposition en 3FN

Le principe d'un tel algorithme consiste à partir d'une couverture minimale des DF à éditer l'ensemble les attributs isolés dans une relation dont tous les attributs sont clés. Ensuite, on recherche le plus grand ensemble X d'attributs qui détermine d'autres attributs A_1, A_2, ….,A_n et l'on sort la relation (X, A_1, A_2, ….,A_n). Une telle relation de clé X est bien en 3FN, les DF
$X \rightarrow A_1, A_2, \ldots,A_n$ sont alors éliminés de la couverture minimale, ainsi que les attributs isolés (non source ou cible de DF) de l'ensemble des attributs.

III.5.5. Notion de forme normale de Boyce-Codd (BCNF)

Une relation est en forme normale de Boyce Codd, si et seulement si, les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé détermine un attribut.
Exemple :
R (matière , N°classe , code_prof)
N'est pas en BCNF, car matière (partie de l'identifiant) dépend de la propriété code_prof

Les langages de manipulation de données relationnelles

IV.1. Introduction

Un langage de manipulation de données se compose d'un ensemble de commandes permettant d'interroger et de modifier une base de données. La modification inclut l'insertion, la mise à jour et la suppression.

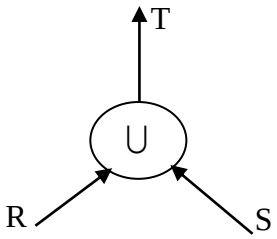
Un langage de manipulation de données n'est pas suffisant à lui seul ; généralement incorporable dans un langage de programmation classique appelé langage hôte afin de réaliser des transactions programmées.
Ainsi, la plupart des SGBD disposent d'un langage de manipulation de données externe conversationnel et proposent une intégration de ce langage dans un langage hôte.

On peut distinguer trois grandes classes de langages : les langages basés sur l'algèbre relationnelle, le langage d'IBM SQL, les langages basés sur la logique des prédicats [25,26].

IV.2. L'algèbre relationnelle

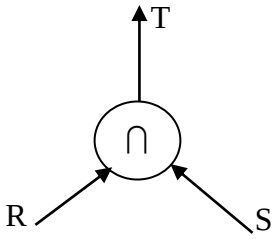
L'algèbre relationnelle a été inventée par IBM-CODD comme une collection d'opérations formelles sur les relations.

L'union
L'union de deux relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples appartenant à R ou S ou aux deux relations.
On notera cette opération : T=R⋃S ou graphiquement [25,26]:



🕮 L'union de deux relations R et S de même structure, est la relation T de même structure, contenant les tuples appartenant dans R ou dans S.

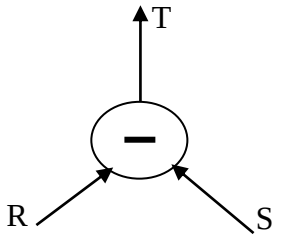
L'intersection
L'intersection de deux relations R et S de même schéma est une relation T de même schéma contenant les tuples appartenant à la fois à R et S.
On notera cette opération : T=R⋂S ou graphiquement [25,26]:



La différence

La différence de deux relations R et S de même schéma (dans l'ordre R-S) est une relation T de même schéma contenant les tuples appartenant à R et n'appartenant pas à S.

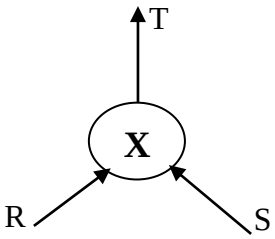
On notera cette opération : T=R-S ou graphiquement [25,26]:



Le produit cartésien

Le produit cartésien de deux relations (de schéma quelconque)R et S, est une relation ayant pour attribut la concaténation des ceux de R et S et dont les tuples sont toutes les concaténations d'un tuple de R à un tuple de S.

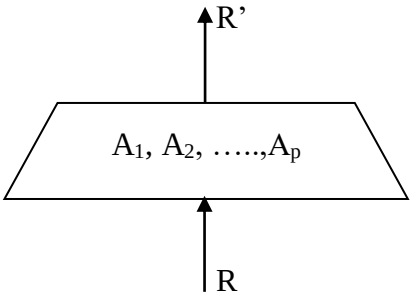
On notera cette opération : T=RxS ou graphiquement [25,26]:



La projection

La projection d'une relation de schéma R(A_1, A_2, ...,A_n) sur les attributs A_1, A_2, ...,A_p est une relation R' de schéma R'(A_1, A_2, ...,A_p) dont les tuples sont obtenus par élimination des valeurs des attributs de R n'appartenant pas à R' et par suppression des tuples en double.

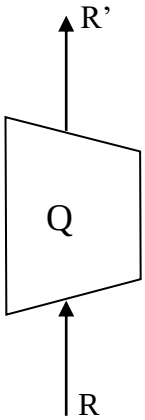
On notera cette opération : R'= Π (A_1, A_2, …..,A_p)[R] ou graphiquement [25-26]:



La sélection (restriction)

L'opération de sélection consiste de sélectionner dans une relation, l'ensemble des tuples satisfaisant une condition donnée appelée 'qualification'.

On notera cette opération : R'=R[Q] ou graphiquement [25-26]:

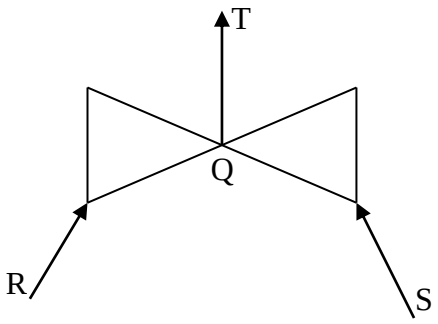


📖 Le résultat d'une sélection sur une relation est une relation de meme structure contenant seulement les tuples vérifiant certaines propriétés, dites 'critères de sélection'
Les critères de sélections sont : <attribut> <opérateur> <valeur>
Les opérateurs sont : <, ≤ , >, ≥, <>, AND, OR, NOT
Exemple : ville = 'Béchar'

La jointure

La jointure de deux relations R et S selon une condition Q est l'ensemble des tuples du produit cartésien RxS satisfaisant la condition Q.

On notera cette opération : T=RxS[Q] ou T=R $\bowtie_Q$S ou graphiquement [25-26]:



On appelle **équi-jointure** une jointure où la sélection se fait selon un critère d'égalité entre un attribut de la première relation et un attribut de la seconde relation.

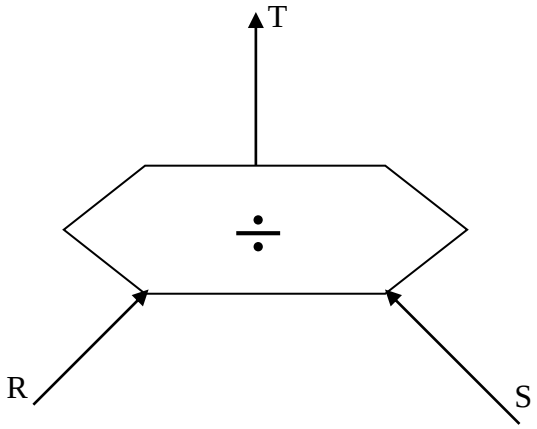
On appelle **Jointure Naturelle**, une équi-jointure sur des attributs portant le même nom dans la première relation et la deuxième relation. La jointure naturelle est l'opérateur de jointure le plus utilisé dans l'algèbre relationnelle.

.

La division

Le quotient de la relation R de schéma R(A_1, A_2,….,A_n) par la sous-relation S de schéma S(A_{p+1},…..,A_n) est la relation T de schéma T(A_1, A_2,…..,A_p) formée de tous les tuples qui concaténés à chacun des tuples de S donne toujours un tuple de R.

On notera cette opération : DIV(R,S) ou T=R ÷ S ou graphiquement [25-26]:



📖 La division de A(a1, a2, a3, a4) par B(a1, a2) est la relation C(a3, a4), telleque le produit cartésien de B par C est contenu dans A

3. Composition d'opérations

Il est possible de composer la plupart des questions que l'on peut poser à une base de données relationnelle, avec les opérations de base successivement enchaînées sur des relations. En effet, les questions peuvent être exprimées à l'aide de succession des opérations : union, différence, jointure, restriction, projection, …etc. La représentation graphique de ces opérations permet de composer des arbres d'opérations relationnelles (voir figure.IV.1).

Arbre Algébrique

Arbre représentant une question dont les nœuds terminaux représentent les relations, les nœuds intermédiaires des opérations de l'algèbre relationnelle, le nœud racine le résultat d'une question, et les arcs les flux de données entre les opérations [24-26].

Une méthode de génération simple d'un arbre consiste à prendre les prédicats de qualification dans l'ordre où ils apparaissent et à leur associer l'opération relationnelle correspondante, puis à terminer par une projection finale pour obtenir le résultat.

Par exemple, soit la base de données composées des relations suivantes :
Etudiant (N°etud, nom, prénom)
Module (N°mod, intitule)
Examen (N°etud, N°mod, note)

Pour répondre à la question suivante ‘nom et prénom de tous les étudiants ayant obtenu une note supérieure à 10 dans le module numéro N’, on peut concevoir l’arbre des opérations suivant :

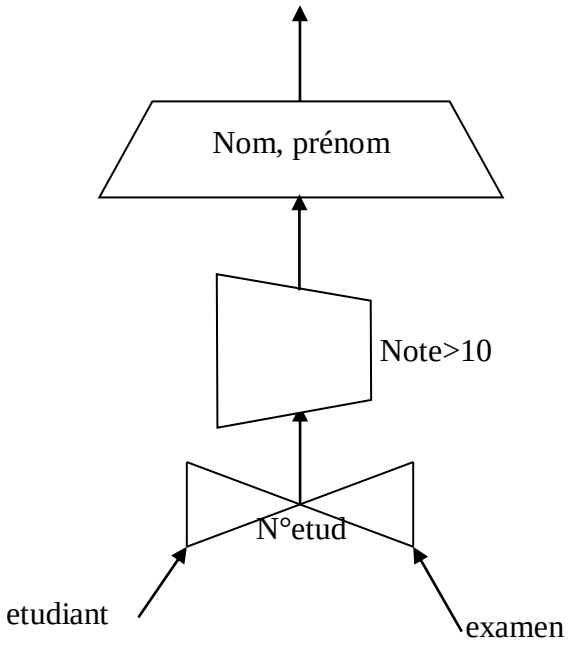


Fig.IV.1.exemple d’arbre d’opérations relationnelles

A partir d’un arbre algébrique, il est possible de générer un plan d’exécution en parcourant l’arbre, des feuilles vers la racine. Une opération peut être exécutée dès que ses opérandes sont disponibles ; ainsi, si l’opération a n’utilise pas les résultats de l’opération b, a et b peuvent être exécutées en parallèle.

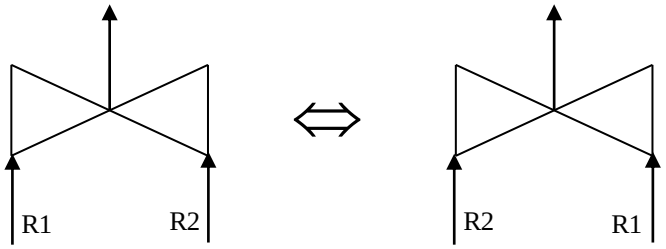
On cherche bien sur à optimiser les temps de réponse, donc à minimiser le temps nécessaire à l’exécution du plan. Le problème est donc de générer un plan optimal. Pour cela, il faut optimiser simultanément :

- Le nombre d’opérations d’entrées-sorties
- Le parallélisme entre les entrées-sorties
- La taille des tampons nécessaires à l’exécution
- Le temps unité central nécessaire

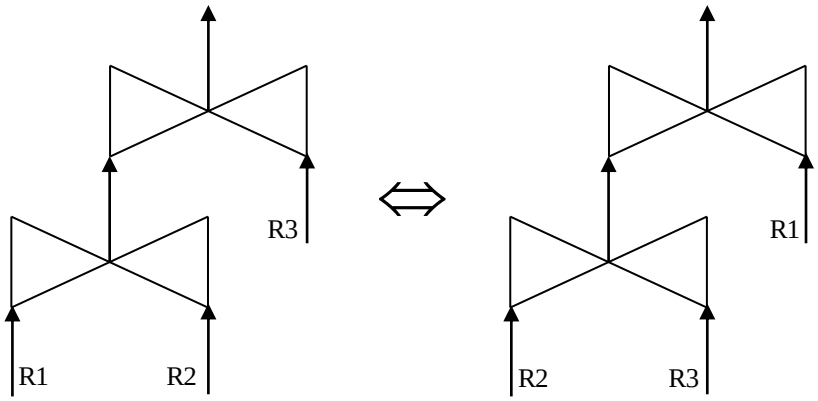
L’optimisation effectuée dépend essentiellement de l’ordre des opérations apparaissent dans l’arbre algébrique utilisé. Il est donc essentiel d’établir des règles permettant de générer, à partir d’un arbre initial, tous les arbres possibles afin de pouvoir ensuite choisir celui conduisant au meilleur plan. Alors, le nombre d’arbres étant très grand, on est amené à définir des heuristiques pour déterminer un arbre proche de l’optimum.

IV.4. Règles de transformation des arbres

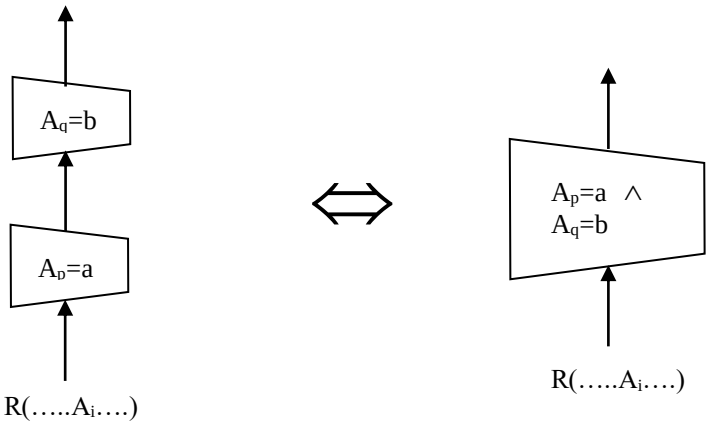
Règle1: Commutativité des jointures



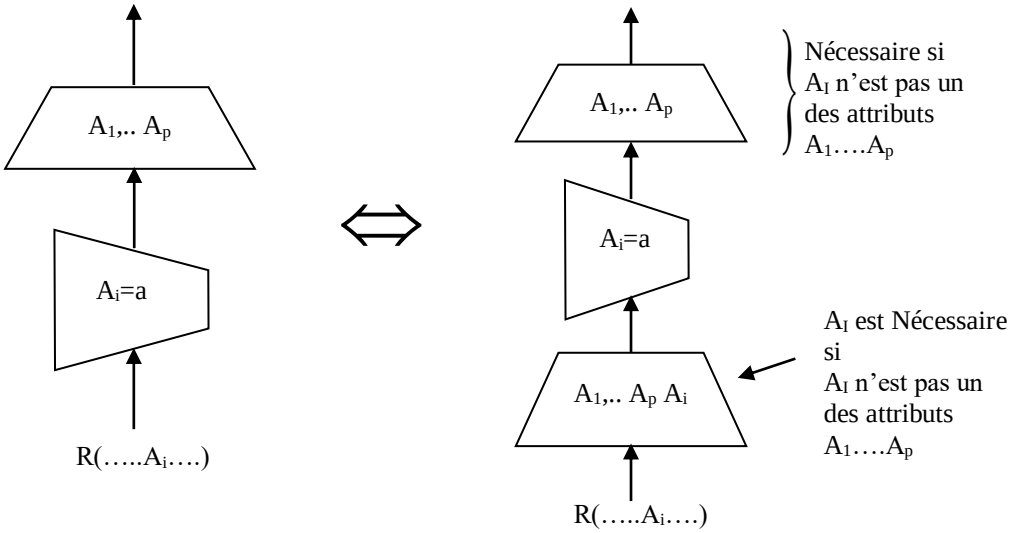
Règle2: Associativité des jointures:



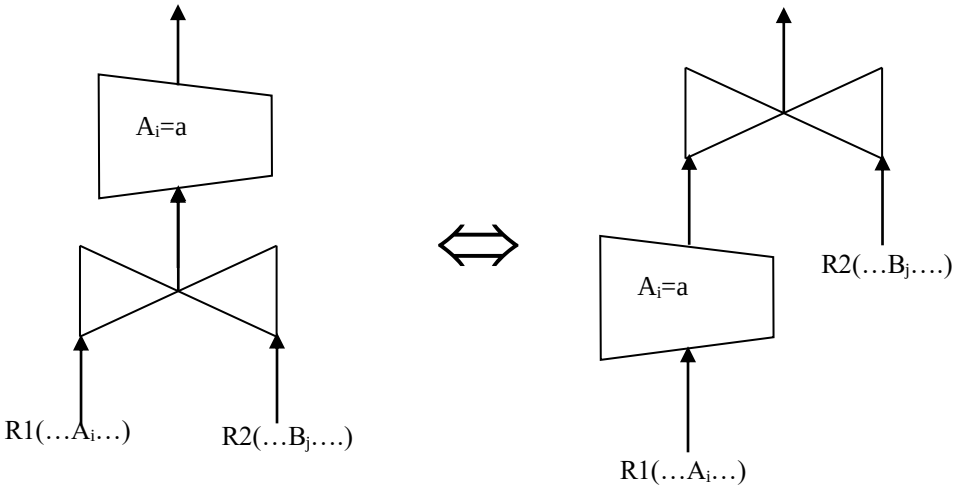
Règle3 : Regroupement des restrictions :



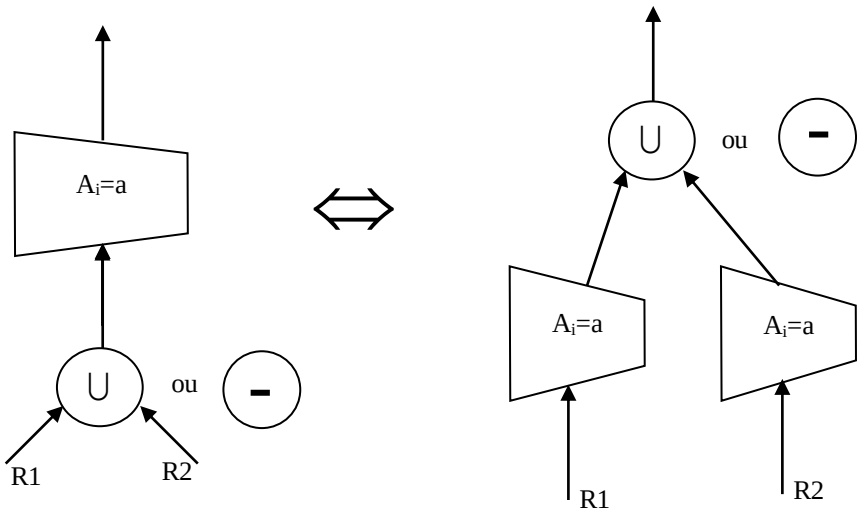
Règle4 : Commutation des restrictions et projections



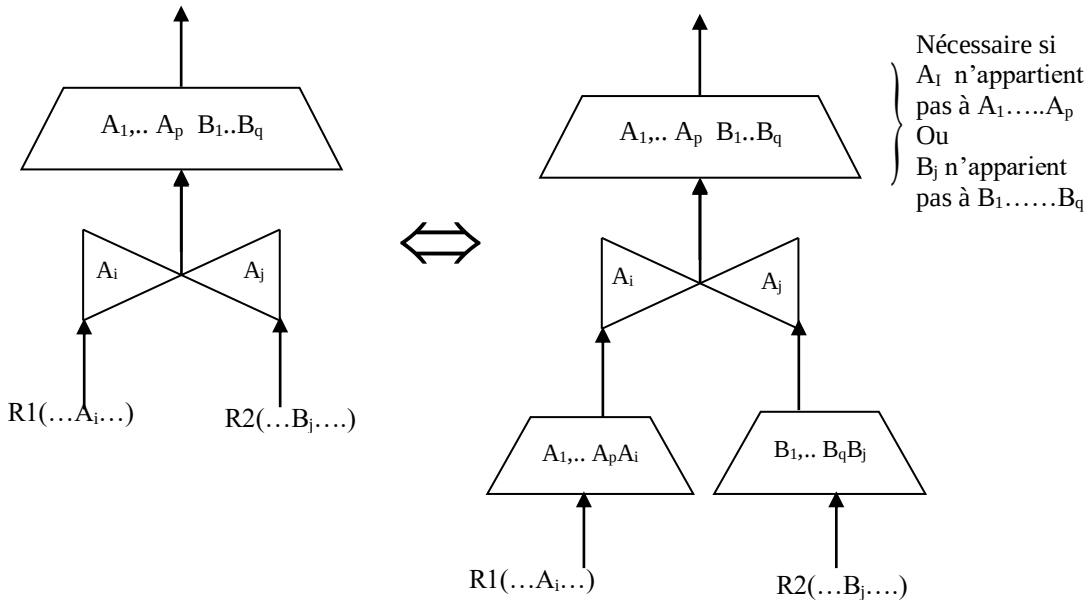
Règle5 : Commutation des restrictions et jointures :



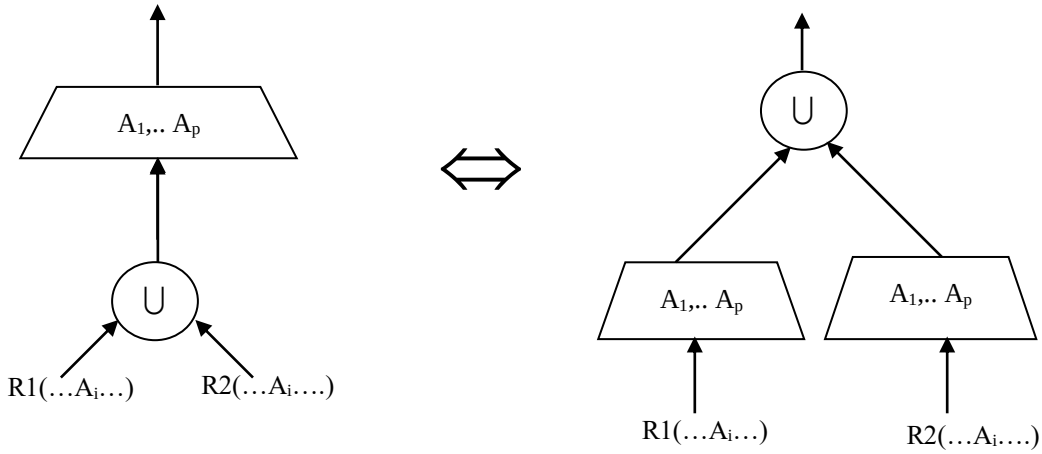
Règle6 : Commutation des restrictions et unions ou des restrictions et différences :



Règle7 : Commutation des projections et jointures :



Règle8 : Commutation des projections et unions :



IV.5. Optimisation par descendante des opérations unaires

Une première optimisation simple consiste à exécuter tout d'abord les opérations unaires (restriction, projection) puis les opérations binaires. En effet, les opérations unaires sont des réducteurs de la taille des relations, alors qu'il n'en est pas ainsi de certaines opérations binaires qui ont tendance à accroître la taille des résultats par rapport aux relations arguments. Aussi, afin de ne considérer que les arbres à flux de données minimum et de réduire ainsi le nombre d'entrées-sorties à effectuer, on est conduit à descendre les restrictions et projections. De plus, quand deux projections successives portent sur une meme relation, il est nécessaire de les regrouper afin d'éviter un double accès à la relation [24-26].

Les principes précédents conduisent à l'algorithme d'optimisation suivant :

1. Séparer les restrictions comportant plusieurs prédicats à l'aide de la règle 3
2. Descendre les restrictions aussi bas que possible à l'aide des règles 4, 5, et 6
3. Regrouper les restrictions successives portant sur une meme relation.
4. Descendre les projections aussi bas que possible à l'aide des règles 7 et 8
5. Regrouper les projections successives en conservant les attributs restants et éliminer d'éventuelles projections inutiles qui auraient pu apparaître (projection sur tous les attributs d'une relation)

En règle générale, une restriction suivie par une projection sont exécutées simultanément par un même opérateur de sélection (restriction + projection) afin d'éviter plusieurs passes sur une même relation. Il en est de même pour une jointure suivie par une projection.

Le langage SQL

Le langage SQL est une évolution commercialisée par IBM du langage SEQUEL initialement développé à IBM San-José comme un langage de recherche. Ce langage peut être perçu comme une expression agréable et très complète de séquences d'opérations relationnelles [27-30].

Nous illustrerons SQL sur les bases de données relationnelles composées des relations suivantes :
client(N°cl, nom_cl, adresse_cl)
article(N°art, nom_art, prix_achat_art, prix_vente_art, N°fourn, poids_art, coul_art)
commande(N°cde, N°cl, date_cde, N°fourn)
ligne_commande(N°cde, N°ligne, N°art, prix_unit, Qte_cde)
fournisseur(N°fourn, nom_fourn, adresse_fourn)

V.1. Structure générale du langage

La forme générale :
SELECT x_1, x_2, …x_p
FROM R
WHERE F
Où x_1, x_2, …x_p sont les noms des attributs, R est le nom d'une relation et F est une expression logique qui utilise les attributs de R.

SELECT permet de rechercher les attributs que l'on souhaite extraire de la base de données
FROM indique dans quelle table (relation) les données sont stockées
Ces deux clauses sont généralement suivies d'une clause **WHERE** pour définir la condition à remplir par les valeurs d'un tuple.

Le résultat d'une clause SELECT est une table, sous-ensemble de la table de départ. Par défaut, la table-résultat s'affiche à l'écran, mais elle peut servir de point de départ pour une autre clause SELECT (requête imbriquée)

V.2. Projection d'une table

Rappelons qu'une projection effectue l'extraction de colonnes (attributs) spécifiées d'une relation puis élimine les tuples en double. Une projection s'exprime à l'aide du langage SQL par la clause :

```
SELECT liste d'attributs
FROM nom de relation
```

Choisir les attributs

La clause SELECT permet de désigner les attributs que l'on veut voir affiché. L'ordre des attributs du résultat sera l'ordre d'apparition dans le SELECT ou l'ordre spécifié dans la structure de la table pour un SELECT

Exemple : pour afficher tous les noms de fournisseurs :

```
SELECT nom_four
FROM fournisseur
```

Empêcher les répétitions de lignes

Par défaut, SQL affiche le résultat d'une requête sans éliminer les répétitions de lignes. Pour le cas contraire, il suffit de rajouter UNIQUE ou DISTINCT dans la clause SELECT.

Alors que la même question sans double s'exprime :

```
SELECT UNIQUE nom_four
FROM          fournisseur
```

L'astérisque * désigne tous les attributs d'une relation :

```
Exemple : SELECT *                ⇔    SELECT  N°cl, nom_cl, adresse_cl
          FROM  client                 FROM    client
```

V.3. Sélection dans une table

V.3.1. les éléments de la clause WHERE

La clause WHERE permet de spécifier un critère de sélection ou prédicat. Si ce prédicat est vérifié par le tuple, celui-ci est retenu. On réalise ainsi l'opération relationnelle de sélection.

Le prédicat est une expression logique composée d'une suite de conditions combinées entre elles par les opérateurs logiques AND, OR ou NOT.
Un élément d'une expression logique peut prendre une des formes suivantes :

- Comparaison à une valeur (=, ≠, <, >, <=, >=)
- Comparaison à un intervalle de valeurs (BETWEEN)
- Comparaison à une liste de valeurs (IN, MATCH)
- Comparaison à une filtre (LIKE)
- Test sur l'indétermination d'une valeur (IS NULL)
- Test 'tous' ou 'au moins un' (ALL, ANY)
- Test existentiel (EXIST)
- Test d'unicité (UNIQUE)

V.3.2. Les comparaisons

Comparaison à une constante

Les chaînes de caractères consistent en une séquence de caractères entre apostrophes, par exemple 'Bonjour', '01234'. Pour inclure l'apostrophe dans une chaîne de caractères, il suffit de la doubler.

- pour retrouver tous les clients habitant à Béchar :

```
SELECT   *
FROM client
WHERE adresse_cl='Béchar'
```

- pour sélectionner tous les articles dont le poids est supérieur à 500g :

```
SELECT   *
FROM article
WHERE poids_art>500
```

Comparaison à une expression

Une expression numérique peut comporter des opérateurs arithmétiques, des noms d'attributs et des constantes.

- pour sélectionner tous les articles pour lesquels le prix de vente est supérieur ou égal au double du prix d'achat :

```
SELECT *
FROM article
WHERE prix_vente_art >= 2*prix_achat_art
```

Comparaison avec l'opérateur AND

Le AND représente le ET logique.

- pour sélectionner tous les articles rouges de poids supérieur à 500g :

```
SELECT *
FROM article
WHERE coul_art='rouge'
AND   poids_art>500
```

Comparaison avec l'opérateur OR

Le OR représente le OU logique.

- Pour sélectionner les articles rouges ou de poids supérieur à 500g :

```
SELECT *
FROM article
WHERE coul_art='rouge'
OR    poids_art>500
```

Comparaison avec l'opérateur NOT

L'opérateur **NOT** inverse la valeur d'une expression logique.

- On souhaite l'inverse de la requête précédente. Pour obtenir les articles qui ne sont pas rouges et dont le poids est inférieur ou égal à 500g :

```
SELECT *
FROM article
WHERE NOT(coul_art='rouge' OR poids_art>500)
```

Comparaison entre deux valeurs

L'opérateur **BETWEEN** permet de sélectionner les tuples dont l'attribut spécifié contient une valeur dans un intervalle précis.

- Pour afficher la liste des articles dont le prix d'achat est compris entre 500 et 1000 DA :

```
SELECT *
FROM article
WHERE prix_achat_art BETWEEN 500 AND 1000
```

- NOT BETWEEN permet de sélectionner les tuples dont un attribut contient une valeur en dehors d'un intervalle.

Comparaison avec une liste de valeurs

L'opérateur **IN** permet de sélectionner les tuples dont l'attribut spécifié contient une valeur appartenant à une liste.

- pour afficher la liste des articles de couleur soit rouge soit vert :

```
SELECT *
FROM article
WHERE coul_art IN ('rouge', 'vert')
```

- NOT IN permet de sélectionner les tuples dont un attribut contient une valeur autre que celle de la liste.

Comparaison avec un filtre pour caractères

L'opérateur **LIKE** permet de sélectionner les tuples dont l'attribut spécifié (de type caractère) contient une valeur correspondant au filtre donné. Avec les opérateurs '=' ou 'IN', la valeur de l'attribut doit correspondre au filtre exactement à la constante qui suit. Deux caractères spéciaux permettent de définir le filtre :

% : n'importe quelle séquence de 0 ou plusieurs caractères.
_ : un seul caractère quelconque.

- pour retrouver un client dont le nom commence par Bend :

```
SELECT *
FROM client
WHERE nom_cl LIKE 'Bend%'
```

- Pour chercher les clients dont le nom commence par Bendj, se termine par ma et il compose de huit lettres :

```
SELECT *
FROM client
WHERE nom_cl LIKE 'Bendj_ma'
```

Comparaison avec un attribut indéterminé

Avec SQL, on utilise la syntaxe **IS NULL** ou inversement IS NOT NULL.

- pour rechercher tous les articles pour lesquels la couleur est indéterminée :

```
SELECT *
FROM article
WHERE coul_art IS NULL
```

V.3.3. Trier les tuples résultats

Il est possible de trier des tuples sélectionnés d'après la valeur d'un ou de plusieurs attributs, ceux-ci doivent obligatoirement faire partie de la liste des attributs dans la clause SELECT. Un index n'est pas obligatoire et SQL crée éventuellement un index temporaire si cela peut optimiser sa recherche. Pour ce faire, utiliser la clause **ORDER BY** suivie du nom des attributs sur lesquels il faut trier. En ajoutant **ASC** ou **DESC**, on précise l'ordre croissant ou décroissant *(par défaut le tri se fait par ordre croissant)*.

- pour trier les articles selon l'ordre croissant de leur poids :

```
SELECT  *
FROM  article
ORDER BY poids_art
```

- lister les numéros, les noms, les poids, et les couleurs des articles triées par ordre croissant de leur poids on aurai :

```
SELECT N°art, nom_art, poids_art, coul_art
FROM  article
ORDER BY poids_art
```

💣 L'expression 'ORDER BY 3' est suffisant, car '3' fait référence au troisième attribut spécifié dans la clause SELECT.

Pour demander un tri avec plusieurs attributs, il suffit de citer leur nom ou leur numéro d'ordre dans la clause ORDER BY, séparés par des vergules.

- pour trier les articles de poids inférieur ou égal à 100g selon l'ordre croissant de leur poids et à poids égal par prix d'achat décroissant :

```
SELECT *
FROM article
WHERE poids_art<= 100
ORDER BY poids_art ASC, prix_achat_art DESC
```

V.3.4. Les fonctions de groupe

Les fonctions de groupe effectuent un calcul sur l'ensemble des valeurs d'un attribut pour un groupe de tuples. Un groupe est un sous-ensemble des tuples d'une table, pour lesquels la valeur d'un attribut reste constante. Les groupes sont spécifiés par la clause **GROUP BY** suivie du nom de l'attribut sur lequel s'effectue le groupement. En l'absence de clause GROUP BY, le groupe est constitué de l'ensemble des tuples sélectionnes.

Les fonctions de groupe classique sont :
COUNT : compte le nombre d'occurrences de l'attribut.
SUM : calcule la somme des valeurs de l'attribut
AVG : calcule la moyenne des valeurs de l'attribut
MAX : recherche la plus grande valeur de l'attribut
MIN : recherche la plus petite valeur de l'attribut

L'argument de la fonction contient le nom de l'attribut sur lequel elle doit être calculée. L'argument précédé de DISTINCT signifie qu'il faut éliminer les répétitions de valeur. Les valeurs indéterminées ne sont jamais prises en compte.

- pour calculer le poids moyen des articles on aurait :

```
SELECT AVG(poids_art)
FROM article
```

- pour calculer le prix de l'article le plus cher du stock on aurait :

```
SELECT MAX(prix_vente_art)
FROM article
```

- pour calculer le poids moyen, la marge maximum, la différence entre le prix de vente maximum et le prix d'achat maximum, pour les articles dont l'attribut coul_art est défini :

```
SELECT AVG(poids_art), MAX(prix_vente_art-prix_achat_art),
        MAX(prix_vente_art)-MAX(prix_achat_art)
FROM  article
WHERE coul IS NOT NULL
```

- pour compter le nombre de couleurs différentes existant dans le stock :

```
SELECT COUNT(DISTINCT coul_art)
FROM article
```

V.3.5. Requête sur les groupes

La clause GROUP BY

La clause GROUP BY divise la table résultant du SELECT en un nombre minimum de groupes, telsque, à l'intérieur de chaque groupe, l'attribut spécifié possède la même valeur pour chaque tuple. Ce n'affecte pas l'organisation physique de la table. Lorsqu'une clause GROUP BY est spécifiée, les fonctions de groupe sont calculées pour chaque groupe.

Cette clause permet d'afficher la valeur de l'attribut commun, suivie de celles de fonctions appliquées à chaque groupe.

La clause HAVING

La clause HAVI NG est l'équivalent du WHERE appliqué aux groupes. Le critère spécifié dans la clause HAVING porte sur la valeur d'une fonction calculée sur un groupe. Les groupes ne répondent pas au critère spécifié dans la clause HAVING ne faut pas partie du résultat.

- pour rechercher la couleur des articles dont le prix de vente moyen des articles de la couleur est supérieur à 100 :

```
SELECT coul_art, AVG(prix_vente_art)
FROM    article
GROUP BY coul_art
HAVING AVG(prix_vente_art)>100
ORDER BY coul_art
```

- pour calculer le prix de vente moyen de chaque couleur d'articles :

```
SELECT coul_art, AVG(prix_vente_art)
FROM article
GROUP BY coul_art
ORDER BY coul_art
```

📖 Les groupes sont crées d'après la valeur de l'attribut coul_art, puis pour chaque valeur de coul_art, SQL calcule la moyenne de l'attribut prix_vente_art.

- pour calculer le prix de vente moyen des articles de chaque couleur en excluant les articles pour lesquels le prix d'achat est inférieur à 50 :

```
SELECT coul_art, AVG(prix_vente_art)
FROM article
WHERE prix_achat_art=>50
GROUP BY coul_art
ORDER BY coul_art
```

V.4. Les sélections sur les tables multiples

SQL permet de réaliser la liaison entre plusieurs tables en utilisant les techniques suivantes [27-30]:

- La jointure réalise la liaison entre deux tables, via l'égalité d'un des attributs, présent dans les deux tables.
- Les requêtes imbriquées permettent d'employer le résultat d'une requête comme élément d'une autre requête.
- Les opérateurs ensemblistes offrent la combinaison des résultats de requêtes

V.4.1. La jointure

Pour écrire une jointure voici quelques conseils :

- Déterminer les tables à mettre enjeu, les inclure dans la clause FROM
- Inclure les attributs à visualiser dans la clause SELECT
- Si la clause SELECT comporte des fonctions sur les groupes, il faut une clause GROUP BY reprenant tous les attributs cités dans la clause SELECT, sauf les arguments des fonctions en question.
- Déterminer les conditions limitant la recherche. Les conditions portant sur les groupes doivent figurer dans une clause HAVING, celles portant sur des valeurs individuelles dans une clause WHERE
- Pour employer une fonction sur les groupes dans une clause WHERE, ou si on a besoin de la valeur d'un attribut d'une autre table, il est nécessaire d'utiliser une requête imbriquée
- Pour fusionner les résultats venant de deux clauses SELECT, il suffit de les joindre par une UNION
- Préciser l'ordre d'apparition des tuples du résultat dans une clause ORDER BY

Exemple : pour sélectionner les articles de couleur 'rouge' et afficher le numéro, le poids de l'article et le nom du fournisseur :

```
SELECT N°art, poids_art, nom_fourn
FROM article, fournisseur
WHERE article.N°fourn = fournisseur.N°fourn
AND coul_art = 'rouge'
```

📖 On pourrait penser, de façon simple, que SQL effectue le produit cartésien des tables citées dans la clause FROM. Ainsi, si la 1ère table comporte N tuples et la 2ème table comporte M tuples, le produit cartésien contient M*N tuples : ceci correspond à une simple vue de l'esprit

En réalité, l'optimiseur va jouer son rôle en effectuant les sélections et les projections sur les tables de départs.

- un cas particulier simple de jointure sans condition est le produit cartésien. Celui-ci s'exprime très simplement en incluant plusieurs relations dans la clause FROM
  exemple : le produit cartésien des relations article et client

```
SELECT *
FROM client, article
```

V.4.2. les requetes imbriquées

Donc pour optimiser la liaison entre plusieurs tables, rien n'empêche d'imbriquer plusieurs requêtes, le résultat de la requête la plus imbriquée servant de table de départ à la requête de niveau immédiatement supérieur.

Il faut s'avoir que le temps nécessaire à obtenir le résultat peut être très diffèrent d'une écriture à l'autre, et qu'il dépend de la logique suivie par l'optimiseur pour organiser la recherche. Il n'est donc pas, à priori, possible de prévoir l'écriture la plus performante.

- pour rechercher tous les articles dont le poids est inférieur au poids de l'article numéro 'A002' :

```
SELECT N°art, poids_art
FROM article
WHERE poids_art < (SELECT poids_art
                   FROM article
                   WHERE N°art='A002')
ORDER BY 2
```

- pour rechercher dans la table article, les articles de même couleur que l'article 'A020' et dont le poids est supérieur au poids moyen de tous les articles :

```
SELECT N°art, nom_art
FROM article
WHERE coul_art = (SELECT coul_art
                  FROM article
                  WHERE N°art='A020')
AND poids_art > (SELECT AVG(poids_art)
                 FROM article)
```

- donner la liste des fournisseurs qui vendent au moins un article de couleur noir

```
SELECT nom_four
FROM fournisseur
WHERE N°_four IN (SELECT N°_four
                  FROM article
                  WHERE coul_art = 'noir')
```

- Si la liste obtenue est traitée par un des opérateurs **ALL** ou **ANY**, la requête est à placer dans une clause WHERE composée d'un opérateur de comparaison (=, <>, <, >, ≤, ≥) suivi d'un des opérateurs ALL ou ANY, et suivi de la requete imbriquée.

  - Avec l'opérateur ALL, la condition est vérifiée si la comparaison se vérifie pour toutes les valeurs ramenées par la requete imbriquée.
  - Avec l'opérateur ANY ou **SOME**, la condition est vérifiée si la comparaison se vérifie pour au moins une des valeurs ramenées par la requete imbriquée.

💣 L'emploi de ANY ou ALL est assez subtil et peut facilement conduire à des erreurs. Il est préférable d'employer à leur place des requetes employant des fonctions sur les groupes ou les requetes existentielles.

- on veut rechercher la liste des articles dont le prix de vente est supérieur au prix de vente de l'article de couleur blanche le moins cher.

```
SELECT nom_art
FROM article
WHERE prix_vente_art > ANY (SELECT prix_vente_art
                            FROM article
                            WHERE coul_art = 'blanc')
```

Autre méthode :

```
SELECT nom_art
FROM article
WHERE prix_vente_art > (SELECT MIN(prix_vente_art)
                        FROM article
                        WHERE coul_art = 'blanc')
```

- On peut utiliser la clause **EXISTS** après la clause WHERE. La condition est vérifiée si la requete imbriquée donne un résultat composé d'au moins un tuple.
  Exemple : pour résoudre l'exemple précédent par une requete existentielle :

```
SELECT nom_art
FROM article a
WHERE EXISTS (SELECT *
              FROM article b
              WHERE b.coul_art = 'blanc'
              AND  a.prix_vente_art > b.prix_vente_art)
```

🕮 Pour chaque tuple de la table article, on réalise la requête existentielle : Si on trouve un article de couleur 'blanc' dont le prix de vente est inférieur, le tuple sera sélectionné.

- La requête est à placer dans une clause WHERE après le mot clé **UNIQUE**. La condition est vérifiée si la requete imbriquée donne un résultat contenant un seul tuple.
  Exemple : pour rechercher la liste des clients qui ont acheté plus d'une fois sur la semaine du 9 au 18 septembre :

```
SELECT N°cl, nom_cl
FROM client
WHERE NOT UNIQUE (SELECT *
                  FROM commande
                  WHERE date_cde BETWEEN '09/09/22'AND'18/09/22')
```

- Une clause HAVING permet de spécifier un critère valable pour les groupes. Ce critère peut lui même dépendre du résultat d'une requête imbriquée.
  Exemple :pour rechercher la liste des articles dont la somme des ventes est supérieur à la moyenne de la somme des ventes de tous les articles :

```
SELECT N°art, SUM(prix_u_art*qte_cde)
FROM ligne_commande
GROUP BY N°art
HAVING SUM(prix_u_art*qte_cde)>(SELECT AVG(prix_u_art*qte_cde)
                                FROM ligne_commande)
```

V.4.3. Les opérateurs ensemblistes

Les opérations ensemblistes sont :
- l'union
- l'intersection
- la différence
- le produit cartésien

- L'opérateur **UNION** effectue l'union des résultats de deux requetes SELECT, c'est à dire qu'à partir des deux tables-résultats, il en crée une troisième comportant l'ensemble des tuples des deux tables de départ, en éliminant les répétitions de tuples.
  En faisant suivre l'opérateur UNION du mot clé ALL, les tuples égaux sont conservés.

- Si les tables unies sont des sous-ensembles de la même table, l'emploi de cet opérateur est tout à fait équivalent à celui du OR placé entre deux conditions.

- L'opérateur **INTERSECT** effectue l'intersection des résultats de deux requêtes SELECT, c'est à dire qu'à partir des deux tables résultats, il en crée une troisième comportant l'ensemble des tuples communs dans les deux tables de départ.

- L'opérateur **EXCEPT** effectue la différence des résultats de deux requêtes SELECT, c'est à dire qu'à partir des deux tables résultats, il en crée une troisième comportant l'ensemble des tuples de la première requete, qui n'apparaissent pas dans la deuxième.

V.5. L'intégrité des données

Il est important de spécifier toutes les contraintes lors de l'établissement du schéma relationnel, et notamment les contraintes d'intégrité référentielle qui assurent la cohérence entre les clés primaires et les clés étrangères [28-31].

V.5.1. Création de domaine

CREATE DOMAIN nom_domaine IS type_de_la_donnée
(EDIT STRING IS format_d'affichage)
(QUERY HEADER IS nom_de_colonne_à_afficher)
Exemple :
CREATE DOMAIN date_dom IS date
EDIT STRING IS 'JJ-MM-AAAA'
QUERY HEADER IS 'en date de'
On a :
Le nom du domaine :date_dom
Le type de domaine : date
Les attributs qui seront de type date_dom seront affichés sous la forme :
2 caractères pour le jour
2 caractères pour le mois
4 caractères pour l'année
A chaque requête SELECT les attributs de type date_dom qui seront affichés auront pour en tête de colonne, le libellé :'en date de'

V.5.2. Création de table

CREATE TABLE nom_de_table
(définition_des_colonnes
(, contraintes_de_table))
Créer une table consiste à :

Définir les colonnes qui la constituent, donc les attributs de la relation
Eventuellement définir les contraintes portant sur la table ou plus exactement sur certains attributs de cette table.

Définir les colonnes consiste à créer une liste de colonnes.
Pour définir une colonne, on trouve :

Le nom de la colonne.
Le type de la colonne ou le nom d'un domaine.
Les mêmes options que pour un domaine, à savoir : DEFAUT, EDIT STRING, QUERY HEADER.
Les options permettant de définir si l'attribut est le résultat d'une fonction mathématique ou d'une fonction de chaînes de caractères : COMPUTED BY suivi de l'expression à évaluer.

Exemples :

1- CREATE DOMAINE domcli IS char(30) ;
CREATE TABLE client
(cli_num Integer, cli_nom domcli,….) ;

on a :
le nom du domaine : domcli
la création de la table client avec entre autres atributs :

- Le numéro du client, cli_num de type entier
- Le nom du client de type domcli

2- CREATE TABLE statut
(statut_code char(3) PRIMARY KEY,…..) ;
on a la création de la table statut avec entre autres l'attribut statut_code sur trois caractères, défini comme clé primaire de la relation.

3- CREATE TABLE condidates
(laste_name char(40) NOT NULL,….) ;
On a la création de la table condidates avec entre autres l'attribut last_name codé sur 40 caractères qui est nécessairement informé par l'utilisateur à l'insertion de chaque tuple (NOT NULL).

4- CREATE TABLE résumes
(employee_id char(5) UNIQUE,…..) ;
on a la création de la table résumes avec entre autres l'attribut employee_id qui est codé sur 5 caractères avec contrôle d'unicité de la valeur. On ne peut rencontrer, dans cette table deux tuples portant la même valeur pour cet attribut.

- Si l'on ne précise rien pour un attribut, il peut être non informé (NULL). Si l'on veut contraindre l'utilisateur à donner une valeur à un attribut, il suffit de définir la contraindre NOT NULL pour ce dernier.
- La contrainte DEFAULT vous permet de donner une valeur par défaut à un attribut.
- La contrainte UNIQUE oblige le SGBD à contrôler l'unicité des valeurs pour cet attribut, au niveau d'une table. Un attribut défini comme clé primaire (PRIMARY KEY) est d'office UNIQUE et NOT NULL.
- La contrainte CHECK permet de définir une valeur citée dans une liste suivant l'opérateur IN.

Exemples :

1- CREATE TABLE…..
(…..,
clt_type char(16) DEFAULT 'particulier'
CHECK (clt_type IN ('particulier', 'administration', 'grand_compte'))
CONSTRAINT type_clients,
……) ;
on a la création d'une table dans laquelle on définit un attribut clt_type avec :

- ♦ Par défaut la valeur 'particulier'
- ♦ S'il est informé, cet attribut ne peut avoir que l'une des valeurs citées dans la liste suivant l'opérateur IN.
- ♦ Cette contrainte porte le nom type_clients.
- ♦ En cas d'erreur le SGBD donnera pour message erreur :violation de la contrainte 'type_clients'

- Lorsque la clé primaire est constituée d'un seul attribut, il suffit de spécifier la clause PRIMARY KEY dans la définition de l'attribut comme dans le code qui suit :

CREATE TABLE……
(N°clt INTEGER PRIMARY KEY,….) ;

Si la clé primaire est constituée de plusieurs attributs, elle ne pourra être définie qu'au niveau des contraintes de table, donc après la définition des diverses colonnes comme ci dessous :

CREATE TABLE assemblage
(artfini char(8) NOT NULL,
artprim char(8) NOT NULL,
assqte Integer NOT NULL,
PRIMARY KEY(artfini, artprim)CONSTRAINT class
……) ;

V.5.3. La sécurité

Toute opération ne peut etre effectuée que si l'utilisateur en a l'autorisation ou le privilège.
Tous les objets manipulables, au niveau d'un BDD, sont définis dans le schéma par le propriétaire de ce dernier. Celui ci a tous les droits et octroie aux autres utilisateurs diverses autorisations suivant les manipulations qu'ils sont susceptibles de pouvoir faire [28-32].

Les clauses GRANT et REVOKE permettent l'octroi ou la suppression de privilèges.
Dès qu'un utilisateur se connecte à une BDD, le SGBD l'identifie en tant que 'USER' possédant un certain profil de privilèges.
🕮 Les privilèges sont SELECT, INSERT, UPDATE, DELETE, CONNECT…….

La clause GRANT

GRANT liste_des_privilèges ON objet TO liste_des_utilisateurs
Exemples :
- GRANT UPDATE, SELECT ON personnel TO Mostefa
  On a Mostefa qui est autorisé à lire et mettre à jour les données de la table personnel
- GRANT UPDATE(N°cli, nom_cli, ville_cli) ON client TO Omar
  On a Omar qui est autorisé à ne mettre à jour que les colonnes N°cli, nom_cli, ville_cli de la table client.
- GRANT SELECT ON departement TO PUBLIC
  On a tous les utilisateurs qui sont autorisés à lire les données de la table departement

La clause REVOKE

REVOKE liste_des_privilèges ON objet FROM liste_des_utilisateurs
Exemples :
- REVOKE UPDATE ON personnel FROM Mostefa
  On a Mostefa qui n'est plus autorisé à modifier la table personnel
- REVOKE UPDATE (prix_u_article) ON article FROM Rafik
  On a Rafik qui n'est plus autorisé à modifier le prix unitaire dans la table article
- GRANT ALL ON client TO Mostefa
  REVOKE DELETE ON client FROM Mostefa
  On a Mostefa qui a tous les droits pour la table client, sauf celui de supprimer des tuples.
- GRANT SELECT ON client TO PUBLIC
  REVOKE SELECT ON client FROM Omar, Rafik
  On a tous les utilisateurs, sauf Omar et Rafik qui peuvent lire les données de la table client.

La clause WITH GRANT OPTION

Ceux qui reçoivent un privilège ne peuvent le donner à d'autres utilisateurs sauf si ce privilège leur a été transmis avec l'option 'WITH GRANT OPTION'
Exemple :
- GRANT SELECT ON diplome TO Rafik WITH GRANT OPTION
  On a Rafik qui est autorisée à lire les données de la table diplome et pourra redistribuer ce privilège à d'autres utilisateurs.

La clause GRANT OPTION FOR

La clause GRANT OPTION FOR permet de retirer à un utilisateur la possibilité de redistribuer un privilège reçu avec l'option 'WITH GRANT OPTION'
Exemple :

- REVOKE GRANT OPTION FOR SELECT ON departement FROM Omar
  On a Omar qui garde la possibilité de lire les données de la table departement mais elle ne pourra plus redistribuer ce privilège.

La clause CASCADE/RESTRICT

Soit les définitions suivantes :
Mostefa permet à Omar de lire la table client :

GRANT SELECT ON client TO Omar WITH GRANT OPTION

Omar à son tour autorise rafik à lire cette table :

GRANT SELECT ON client TO Rafik WITH GRANT OPTION

Supposons que Mostefa retire le privilège à Omar, sans autre option rafik garde son privilège.
Avec l'option CASCADE, les deux privilèges disparaitssent ainsi que tous les privilèges dirivés :

REVOKE SELECT ON client FROM Omar CASCADE

Avec l'option RESTRICT l'instruction est refusée à cause de la présence d'un privilège dérivé :

REVOKE SELECT ON client FROM Omar RESTRICT

V.5.4. Création une vue

Le concept de vue permet de donner à l'utilisateur une perception simplifiée de la BDD, en ne lui offrant que ce dont il a besoin. Elle apporte une réponse au souci de confidentialité de l'information. La création d'une vue est associée à un SELECT et peut faire l'objet de modifications, sous certaines conditions. On peut imaginer une vue comme une sorte de fenêtre par laquelle il est possible de visualiser ou éventuellement, de modifier l'information contenue dans la table.
La syntaxe est la suivante :

```
CREATE VIEW nom_de_la_vue(liste_des_colonnes)
AS expression_select
```

Exemples :

```
CREATE VIEW client_5000
AS SELECT nom, adresse, chiffre FROM client WHERE chiffre>5000
```

On a la vue client_5000 qui aura pour colonnes nom, adresse et chiffre, c'est à dire par défaut les colonnes citées au niveau du SELECT.

Vue et mises à jour

Soit la vue suivante :

```
CREATE VIEW clients_Béchar(ref_cli, nom, loc, nb_cde)
AS  SELECT N°cl, nom_cl, adresse_cl, cde_cl FROM client
WHERE adresse_cl = 'Béchar'

UPDATE clients_Béchar
SET nb_cde=0
WHERE ref_cli > 600
```

L'opération de mise à jour se transforme en fait comme ceci :

```
UPDATE client
SET cde_cl=0
WHERE  adresse_cl='Béchar'
AND N°cl > 600
INSERT INTO clients_Béchar VALUE(200, 'Mostefa', 'Béchar', 09)
```

Privilèges et vues

En combinant privilèges et vues, on atteint une grande souplesse de sécurité des données

```
CREATE VIEW article_revient
AS SELECT N°art, nom_art, prix_vente_art, coul_art
FROM article
```

- GRANT SELECT, UPDATE ON article_revient TO Mostefa, Omar
- REVOKE SELECT UPDATE, DELETE, INSERT ON article FROM Rafik

💣 La vue crée est parfaitement modifiable et on autorise les utilisateurs à y accéder en lecture et en mise à jour. Ceci fait, on leur interdit alors l'accès à la table permanente pour limiter tout risque de perte d'information.

Exercices sur les modèles de données

Exercice 01

Soit le schéma relationnel :
R(*A*,*B*,*C*,*D*) et **F**={$A \rightarrow B$, $B \rightarrow C$, $D \rightarrow B$}.
Trouver la clé ?

Exercice 02

Soit le schéma relationnel :
R(*Cours*, *Professeur*, *Horaire*, *Salle*, *Etudiant*, *Note*)
F={$C \rightarrow P$, $HS \rightarrow C$, $HP \rightarrow S$, $CE \rightarrow N$, $HE \rightarrow S$}
Trouver la clé?

Exercice 03

La relation suivante donne deux décompositions possibles :

Voiture	N°V	Marque	Type	Puissance	Couleur
	872RH75	Renault	R12TS	6	Bleue
	975AB80	Renault	R12TS	6	Rouge

Décomposition1 :

R1	N°V	Type	Couleur
	872RH75	R12TS	Bleue
	975AB80	R12TS	Rouge

R2	Type	Marque	Puissance
	R12TS	Renault	6

Décomposition2 :

V1	N°V	Type
	872RH75	R12TS
	975AB80	R12TS

V2	Type	Puissance	Couleur
	R12TS	6	Bleue
	R12TS	6	Rouge

V3	Type	Marque
	R12TS	Renault

Quelle est la décomposition sans perte des informations ?

Exercice 04

Soient deux relations R1 et R2 dont les schémas de relations sont :
U1={N°piece, prix_unit, taux_tva, libellé, catégorie}
F1={N°piece $\rightarrow$prix_unit ; N°piece $\rightarrow$catégorie ; N°piece $\rightarrow$libellé ; N°piece $\rightarrow$taux_tva ; catégorie $\rightarrow$taux_tva}
U2={N°gamme, nom_gamme, N°opéra, rang_opéra, nom_opéra}
F2={N°gamme $\rightarrow$nom_gamme ; N°opéra $\rightarrow$nom_opéra ;
N°gamme,N°opéra $\rightarrow$rang_opéra}

1- Quelle sont les clés des relations R1 et R2
2- F1 et F2 constituent une couverture minimale ?
3- En quelle forme normale sont R1 et R2
4- tracez le graphe de dépendance fonctionnel de F1 et F2

Exercice 05

Soit la relation :
maison(ville, rue, N°maison, contenu_maison, population_ville)

1- Déterminer la clé
2- Déterminer la FN de la relation
3- La relation est en 3ème FN ? sinon normaliser-la ?

Exercice 06

Pour constituer une base de données sur la scolarité des étudiants, on dispose des éléments suivantes :
N°etud, N°mod, nom_mod, nbre_heure_enseignement, N°enseig, nom_enseig, grade, indice, nbre_enseig_dép, N°dép, résultat_mod, nom_etud
On fait les hypothèses suivantes :

- Résultat module caractérise un étudiant pour un module donnée
- Chaque module n'est assuré que par un enseignant
- Nombre des heures enseignement est spécifique à un module
- Un enseignant est rattaché à un seul département
- A chaque grade correspond un indice
- Chaque département correspond un nombre des enseignants

1- Déterminez la clé de la relation R
2- Déterminez les DF de R
3- Déduire la FN de R
4- Trouvez la couverture minimale des DF
5- Trouvez la fermeture transitive de l'ensemble des DF

Exercice 07

Soit la relation universelle R<U,F> :
L'ensemble des attributs : U=[N°interv, type_interv, N°contra, type_contra, mont_comm]
L'ensemble des dépendances :
F=[N°contra $\rightarrow$type_contra ; N°interv $\rightarrow$type_interv ; N°contra,N°interv $\rightarrow$mont_comm]
Considérons l'occurrence de relation suivante :

N°interv	type_interv	N°contra	type_contra	mont_comm
27	Courtier	C_52	Court	1500
35	Agent	C_52	Court	1800
40	Courtier	C_52	Court	1200
27	Courtier	C_13	Long	1800
35	Agent	C_13	Long	1300
40	Courtier	C_48	Moyen	1000

1- Déterminer la clé de la relation R
2- En quelle forme normale est la relation R ?
3- Considérons les projections suivantes :
   R1= Π (N°contra, type_contra, type_interv)[R]
   R2= Π (N°interv, type_interv, mont_comm)[R]
   Trouver R1 et R2, et calculer la jointure de R1 et R2. Que constatez-vous ?
4- Montrer formellement que ce résultat était prévisible
5- Utiliser la méthode par décomposition pour normaliser la relation initiale(mettre en 3FN)
6- Considérons les deux projections suivantes :
   R3= Π (N°contra, N°interv, type_interv, mont_comm)[R]
   R4= Π (N°contra, type_contra)[R]
   Montrer que la jointure de cette décomposition conduit à la relation initiale.

Exercice 08

Soit la relation R1 suivante :
R1(cours, section, salle, taille_salle, heure)
1- donner la clé primaire de la relation
2- La relation répond elle à la 2FN ?
3- R1 est elle en 3FN ? Sinon normaliser-la ?

Exercice 09

On considère la relation suivante décrivants des voitures
R(N°imm, puissance, marque, pays, agence, chiffre_affaire)
On fait les hypothèses suivantes :
- chaque véhicule est caractérisé par une puissance et une marque
- une marque est spécifique d'un pays
- le chiffre d'affaire faire référence à une agence pour une marque donnée

1- Quelle est la clé de la relation
2- En quelle FN est cette relation
3- Donnez la couverture minimale de DF et tracez le graphe

Exercice 10

Soit la relation suivante :
prod_fourn(N°prod, N°fourn, nom_fourn, désignation_prod, prix_achat, prix_vente, N°acheteur, nom_acheteur, taux_reduction)
1- déterminer la clé de la relation
2- déterminer les DF de la relation
3- cette relation est en 3ème FN ? sinon normaliser-la ?

Exercice 11

Une agence de voyages offre des voyages organisés dans différentes villes et veut gérer ses informations avec un SGBD relationnel. Voici trois tables définies dans le schéma relationnel pour cette application :
hôtels (nom_hot, ville_hot, lits_disp, date) : cette table contient pour chaque hôtel, identifié par son nom, la ville où il se trouve et le nombre de lits disponibles à une date donnée.
vols (N°vol, compagnie, ville_dep, ville_arr, date_dep, date_arr, places_disp) : cette relation contient pour chaque vol, identifié par son numéro, sa compagnie, ses villes et dates de départ et d'arrivée et le nombre de places disponibles.
voyages (N°voyage, N°vol_all, N°vol_ret, nom_hot) : les voyages sont identifiés par un numéro et on connaît les numéros de vols aller et retour et le nom de l'hôtel réservé.

1. Traduisez les contraintes suivantes en dépendances fonctionnelles
   a. Il ne peut pas y avoir deux hôtels avec le même nom dans deux villes différentes.
   b. Pour un hôtel et une date données on connaît le nombre de lits disponibles.
   c. Connaissant le numéro de vol, on connaît la compagnie, la ville de départ et la ville d'arrivée.
2. Donnez la clé de la table hôtels.
3. Montrez par un exemple que la table hôtels n'évite pas certaines anomalies.

Exercice 12

Une entreprise comprend différents services, chacun étant caractérisé par un numéro (N°serv), un nom (nom_serv) de service, le numéro (N°resp) et le nom (nom_resp) de son responsable, pour chaque service il y'a un seul responsable. Un budget (bud_serv) est attribué à un service.

Chaque service gère un ou plusieurs projets, mais un projet est géré par un seul service. Un projet est caractérisé par un numéro (N°proj) et un nom (nom_proj). Un budget (bud_proj) est attribué à un projet.

Les employés de l'entreprise sont affectés à un seul projet. Un employé est caractérisé par un numéro (N°emp) et un nom (nom_emp). Chaque employé peut etre joint par l'intermédiaire d'un numéro de téléphone (N°tel). Un numéro de téléphone peut etre partagé entre plusieurs employés.

Un employé est installé dans un bureau caractérisé par un numéro (N°bur). Un bureau peut accueillir plusieurs employés. La localisation d'un bureau est repérée par le nom de son bâtiment (nom_bat). Un bureau est rattaché pour gestion à un seul service.
1. déterminer les DF de R
2. déterminer la clé de R
3. R est elle en 3FN ? sinon normalisé-la ?

Exercice (supplémentaire)

Soit le schéma relationnel **R**(*A,B,C,D*), **F**={$A \rightarrow B$, $C \rightarrow D$} et la décomposition Delta = {*AB, CD*}.

1. Montrer que Delta n'est pas une décomposition sans perte d'information.
2. Donner une décomposition sans perte d'information.

Exercice (supplémentaire)

Soit le schéma
R (*A,B,C,D*)
F={$A \rightarrow B$, $B \rightarrow C$, $D \rightarrow B$}
et la décomposition :Delta = (*ACD, BD*)
Est-ce que les relations (*ACD*) et (*BD*) obtenues sont en 3FN?

Exercice (supplémentaire)

Est-ce que le schéma
R(*Rue, Ville, CodePostal*)
F={$RV \rightarrow C$, $C \rightarrow V$ }

est en 3FN?

Exercice (supplémentaire)

Montrer que les schémas suivants ne sont pas en 3FN:
Schéma 1 :
R(*A,B,C,D*)
F={$AB \rightarrow C$, $B \rightarrow D$, $BC \rightarrow A$ }
Schéma 2 :
R(*A,B,C,D*)
F={$A \rightarrow B$, $B \rightarrow C$, $A \rightarrow D$, $D \rightarrow C$ }
Schéma 3 :
R(*C,P,H,S,E,N*)
F={$C \rightarrow P$, $HS \rightarrow C$, $HP \rightarrow S$, $CE \rightarrow N$, $HE \rightarrow S$}
Schéma 4 :
R(*F,A,N,P*)
F = {$F \rightarrow A$, $FN \rightarrow P$}
Schéma 5 :
R(*M,A,D,R*)
F = {$MA \rightarrow D$, $MD \rightarrow R$ }

Exercice (supplémentaire)

Soit la relation **R** (N°projet, produit, nom_fournisseur, adresse_four, nombre_pièce)
Nombre_pièce :nombre des pièces approvisionnées par un fournisseur pour un projet et pour un produit.

R	N°projet	Nom_fournisseur	Produit	Adresse_four	Nombre_pièce
	Pr1	Mostefa	X1	Y1	10
	Pr1	Mostefa	X2	Y2	10
	Pr2	Omar	X3	Y3	50
	Pr3	Ahmed	X4	Y4	100
	Pr3	Omar	X1	Y3	30
	Pr2	mostefa	X2	Y1	20

1. Déterminer la clé de R
2. Déterminer les dépendances fonctionnelles
3. Déterminer la forme normale de R
4. Normaliser R
5. Donnez la couverture minimale de dépendance fonctionnelle
6. Déduire la fermeture transitive de l'ensemble des DF et tracez le graphe.

Exercice (supplémentaire)

Soit la relation suivante :
Employé (N°emp , nom_emp , adresse , catégorie , salaire)
Avec les dépendances fonctionnelles suivantes :
N°emp ⟶ nom_emp , adresse , salaire , catégorie
catégorie ⟶ salaire

1. Quelle est la clé primaire de la relation employé
2. Dans quelle forme normale est la relation employé
3. Décomposer la en 3FN si elle ne l'est pas déjà.

Exercice (supplémentaire)

Soient les dépendances fonctionnelles suivantes :
A ⟶ B
A ⟶ C
A,X ⟶ C
B,X ⟶ D
Parmi ces DF, citez seulement les DF élémentaires.

Solutions des exercices sur les modèles de données

Sol. Exercice 01

A et D n'apparaissent dans aucun membre droit de dépendance et font partie de toute clé
Donc la clé est AD

Sol. Exercice 02

La clé est HE

Sol. Exercice 03

R1⋈ R2	N°V	Type	Couleur	Marque	Puissance
	872RH75	R12TS	Bleue	Renault	6
	975AB80	R12TS	Rouge	Renault	6

Donc la décomposition1 permet de retrouver toutes les informations(les tuples) par jointure c'est une décomposition sans perte.

V1⋈V2⋈V3	N°V	Type	Puissance	Couleur	Marque
	872RH75	R12TS	6	Bleue	Renault
	872RH75	R12TS	6	Rouge	Renault
	975AB80	R12TS	6	Bleue	Renault
	975AB80	R12TS	6	Rouge	Renault

La jointure des relations V1, V2 et V3 est différente de la relation initiale, donc c'est une seulement une décomposition.

Sol. Exercice 04

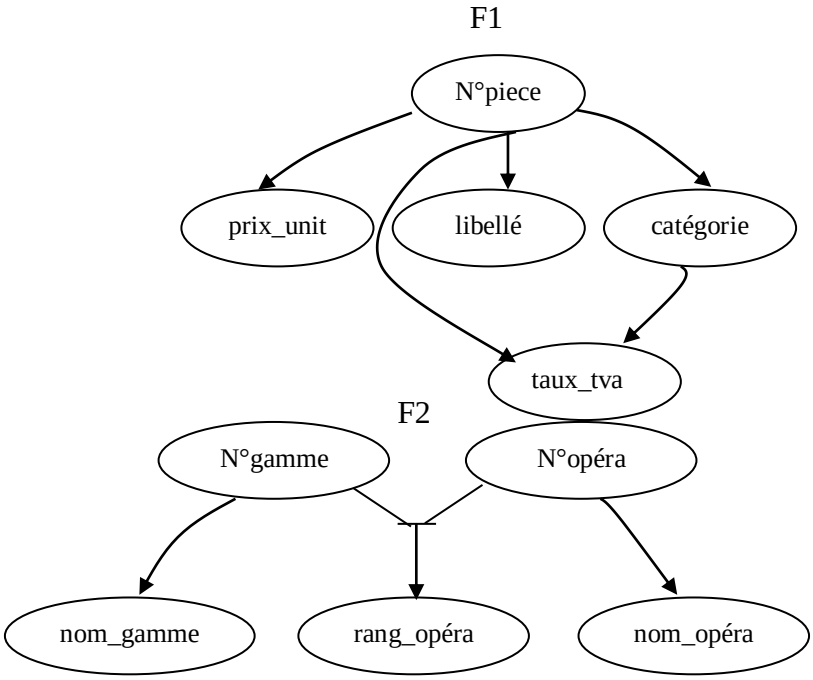
R1=<U1,F1>, R2=<U2,F2>

1- Clé pour U1=N°piece
U2=N°gamme,N°opéra

2- F1 ne constitue pas une couverture minimale car :
N°piece $\rightarrow$taux_tva obtenu par transitivité à partir de N°piece $\rightarrow$catégorie et catégorie $\rightarrow$taux_tva
F2 est une couverture minimale

3- R1 est en 2FN
R2 est en 1FN

4-



Sol. Exercice 05

1- Clé : (ville, rue, N°maison)
DF : ville, rue, N°maison $\rightarrow$contenu_maison
ville $\rightarrow$population_ville
2- Il existe un attribut population_ville dépend fonctionnellement d'une partie de la clé.
ville $\rightarrow$ population_ville
la relation n'est pas en 2FN donc elle est en 1FN
4- La relation n'est pas en 3FN
alors on éclatant la relation :
maison(ville, rue, N°maison, contenu_maison)
ville_pop(ville, population_ville)
elle est en 3FN

Sol. Exercice 06

1- Clé : (N°etud, N°mod)
2- N°etud, N°mod $\rightarrow$resultat_mod
N°mod $\rightarrow$N°enseigant
N°mod $\rightarrow$nbre_heure_enseignement
N°enseig $\rightarrow$N°dep
grade $\rightarrow$indice
N°dep $\rightarrow$nbre_enseignant_dep
N°etud $\rightarrow$nom_etud
N°mod $\rightarrow$nom_mod
N°enseig $\rightarrow$grade
N°enseig $\rightarrow$nom_enseig

3- Il existe des attributs déduit à partir d'une partie de la clé, R n'est pas en 2FN alors R est en 1FN
4- Les DF représentent une couverture minimale, il n'existe pas des dépendances de la forme $X \rightarrow Y$; $Y \rightarrow Z$; et $X \rightarrow Z$ déduite par transitivité
5- $F^{+}=F \cup$ {N°mod $\rightarrow$N°dep ; N°enseig $\rightarrow$indice ; N°mod $\rightarrow$indice ; N°mod $\rightarrow$nom_mod ; N°mod $\rightarrow$nbre_enseignant_dep ; N°ens $\rightarrow$nbre_enseignant_dep}

Sol. Exercice 07

1- La clé : (N°contra, N°interv)
2- 1FN
3- R1

N°contra	type_contra	type_interv
C_52	Court	Courtier
C_52	Court	Agent
C_13	Long	Courtier
C_13	Long	Agent
C_48	Moyen	Courtier

R2

N°interv	type_interv	mont_comm
27	Courtier	1500
35	Agent	1800
40	Courtier	1200
27	Courtier	1800
35	Agent	1300
40	Courtier	1000

R1 ⋈ R2

N°interv	type_interv	N°contra	type_contra	mont_comm
27	Courtier	C_52	Court	1500
40	Courtier	C_52	Court	1200
27	Courtier	C_52	Court	1800
40	Courtier	C_52	Court	1000
35	Agent	C_52	Court	1800
35	Agent	C_52	Court	1300
27	Courtier	C_13	Long	1500
40	Courtier	C_13	Long	1200
27	Courtier	C_13	Long	1800
40	Courtier	C_13	Long	1000
35	Agent	C_13	Long	1800
35	Agent	C_13	Long	1300
27	Courtier	C_48	Moyen	1500
40	Courtier	C_48	Moyen	1200
27	Courtier	C_48	Moyen	1800
40	Courtier	C_48	Moyen	1000

4- On a :
R1(N°contra, type_contra, type_interv)
type_interv ne dépend pas fonctionnellement de N°contra
R2(N°interv, type_interv, mont_comm)
Mont_comm ne dépend pas fonctionnellement de N°interv

5- **R1**(N°contra, N°interv, mont_comm)
R2(N°contra, type_contra)
R3(N°interv, type_interv)

6- R3

N°contra	**N°interv**	**type_interv**	**Mont_comm**
C_52	27	Courtier	1500
C_52	35	Agent	1800
C_52	40	Courtier	1200
C_13	27	Courtier	1800
C_13	35	Agent	1300
C_48	40	Courtier	1000

R4

N°contra	type_contra
C_52	Court
C_13	Long
C_48	Moyen

R3$\bowtie$ R4 donne la relation initiale

Sol. Exercice 08

1- Clé primaire : cours, section
2- R1 est en 2FN car tous les attributs dépendent entièrement de la clé et non pas d'une partie
3- R1 n'est pas en 3FN car il existe un lien entre salle et taille de la salle, ce lien va crée une transitivité

$$\left.\begin{array}{l} \text{cours, section} \rightarrow \text{salle} \\ \text{cours, section} \rightarrow \text{taille\_salle} \\ \text{salle} \rightarrow \text{taille\_salle} \end{array}\right\} \text{ transitivité}$$

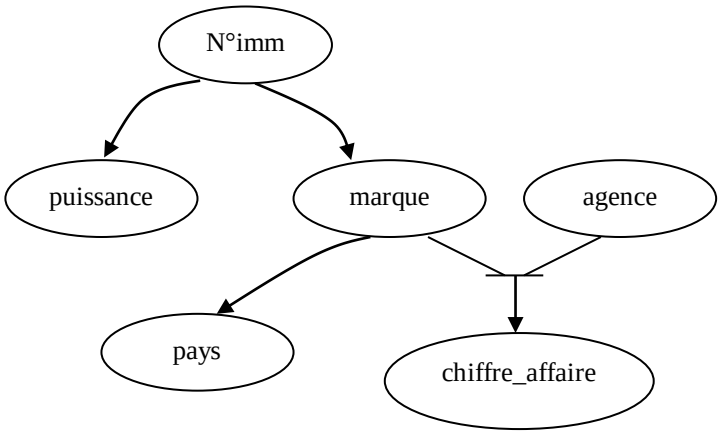
La normalisation est comme suit : supprimer le lien entre cours, section et taille_salle
R1(cours, section, salle, heure)
R2(salle, taille_salle)

Sol. Exercice 09

N°imm → puissance
N°imm → marque
marque → pays
agence, marque →chiffre_affaire
A partir de DF N°imm → marque on utilise l'opération pseudo-transitivité on obtient la dépendance : agence, N°imm → chiffre_affaire
Alors la clé est :N°imm, agence

la relation est en 1FN
la relation n'est pas en 2FN car il existe N°imm → marque, puissance
on éclatant R en
voiture(N°imm, puissance, marque, pays)
agence_affaire(agence, N°imm, chiffre_affaire)
La relation n'est pas en 3FN : il existe DF marque →pays

On éclatant voiture
voiture(N°imm, puissance, marque)
marque_pays(marque, pays)



Sol. Exercice 10

1- la clé est (N°prod, N°fourn)
2- N°prod, N°fourn →tous les attributs
3- La relation est en 1FN
   Elle est en 2FN ? n'est pas en 2FN
   On éclatant la relation :
   fournisseur(N°fourn, nom_fourn)
   produit(N°prod, désignation_prod, prix_vente, N°acheteur, nom_acheteur)
   achat(N°prod, N°fourn, prix_d'achat, taux_réduction)
   La relation produit n'est pas en 3FN car : N°acheteur →nom_acheteur
   On va éclaté produit en deux relations :
   produit(N°prod, désignation_prod, prix_vente, N°acheteur)
   acheteur(N°acheteur, nom_acheteur)

Sol. Exercice 11

1.
   a. nom_hot ⟶ ville_hot
   b. nom_hot , date ⟶ lits_disp
   c. N°vol ⟶ compagnie, ville_dep, ville_arr

2. la clé de la table hôtels est : (nom_hot , date)

Sol. Exercice 12

1. N°serv ⟶ N°resp
   N°serv ⟶ bud_serv
   N°proj ⟶ N°serv
   N°proj ⟶ bud_proj
   N°emp ⟶ N°proj
   N°emp ⟶ N°tel
   N°emp ⟶ N°bur
   N°bur ⟶ nom_bat
   N°bur ⟶ N°serv

   N°serv ⟶ nom_serv
   N°resp ⟶ nom_resp
   N°proj ⟶ nom_proj
   N°emp ⟶ nom_emp

2. la clé :N°emp
3. **R** est en 2FN
   Donc la normalisation est :
   R1(<u>N°emp</u>, N°proj, N°tel, N°bur, nom_emp)
   R2(<u>N°proj</u>, N°serv, bud_proj, nom_proj)
   R3(<u>N°bur</u>, nom_bat, N°serv)
   R4(<u>N°serv</u>, N°resp, bud_serv, nom_serv)
   R5(<u>N°resp</u>, nom_resp)

Exercices sur le langage algébrique

Exercice 01

1- Soit les deux relations suivantes :

R	X1	X2	X3	X4
	100	a	74	12
	110	b	78	13
	120	c	77	12

S	X1	X2	X3	X4
	100	a	74	12
	200	d	79	11

Calculer R $\cup$ S et R-S

2- Soit les deux relations suivantes :

R	X1	X2	X3	X4
	110	a	74	12
	120	b	78	13

S	Y1	Y2	Y3
	Mostefa	i	k
	Rafik	j	t

Calculer le produit cartésien des deux relations R et S

3- Soit la relation suivante :

R	X1	X2	X3	X4
	100	a	74	12
	110	b	77	12
	120	c	77	12
	130	d	78	13

- Calculer la projection de R sur les attributs X3 et X4
- Sélectionner la relation R par la condition X3=77
- Sélectionner la relation R par la condition X3=77 et X4>13

4- Quel est le quotient de la relation R par la relation S

R	X1	X2	X3
	Mostefa	77	12
	Mostefa	79	14
	Mostefa	80	12
	Omar	77	12
	Omar	79	14
	Rafik	79	14

S	X2	X3
	77	12
	79	14

Exercice 02

1- Soient les relations suivantes :

R	**X**	**Y**	**Z**
	x	y	z
	z	y	t
	z	k	t

S	**X**	**Y**	**Z**
	y	w	x
	t	x	k

Calculer R⋃S et R-S

2- Soient les relations suivantes :

R	**X**	**Y**	**Z**
	x	y	z
	t	x	k
	z	y	t

S	**U**	**V**	**W**
	y	g	x
	t	x	k

Calculer RxS

Projection de R sur (X,Y)

Sélection (restriction) de R pour Y=y

Exercice 03

vehicule(N°V, type, marque, puissance, couleur)
personne(N°SS, nom, prenom, adresse)
possede(N°SS, N°V, date, prix)

1- Quel sont les noms des propriétaires de véhicule de type R5 ?
2- Donner le type et la couleur des véhicules de personne habitant à Béchar dont le prénom est M^{ed}
3- Donner la liste des noms des personnes ayant achetées des véhicules de marque Renault entre le 01-01-22 et 10-10-22 et dont le prix est inférieur à 500000DA

Exercice 04

Soit la base de données suivante :
professeur(N°prof, nom_prof, prenom_prof, ville_prof)
stagiaire(N°stag, nom_stag, prenom_stag, origine_stag, ville_stag)
salle(N°salle, capacite_salle)
theme(nom_theme)
stage(N°stage, date_debut, date_fin, ref_cours, N°prof, N°salle)
competance(N°prof, ref_cours)
inscription(N°stage, N°stag)
cours(ref_cours, libelle, duree)

1- Donner les informations sur les stagiaires de la ville de Béchar
2- Donner la date de début et la date de fin du stage N°35
3- Donner la liste des prof qui ont amené de stage N°10
4- Quelle sont les salles libres le 10-01-2001 ?

Exercice 05

Soit le modèle relationnel suivant relatif à une base de données sur des représentations musicales :
representation(N°rep, titre_rep, lieu)
musicien(nom, N°rep)
programmer(date, N°rep, tarif)

1- Donner la liste des titres des représentations
2- Donner la liste des titres des représentations ayant lieu à l'opéra 'Bastille'
3- Donner la liste des noms des musiciens et des titres des représentations aux quels ils participent
4- Donner la liste des titres des représentations, les lieux et les tarifs pour la journée du 09-09-01

Exercice 06

etudiant(N°etud, nom_etud, adresse_etud)
inscription(N°etud, code_mod, date_inscrip)
module(code_mod, nom_mod)
enseignant(N°ens, nom_ens, adresse_ens)
enseignement(N°ens, code_mod)
emploi_temps(N°ens, code_mod, salle, heure, jour)
exprimer par l'intermédiaire des opérateurs de l'algèbre relationnel les requêtes suivantes :

1- Quel sont les noms des étudiants qui suivent des cours avec l'enseignant 'Bendjima' ?
2- Donner les noms des étudiants qui suivent au moins un cours avec l'étudiant 'Omar'
3- Donner les noms des étudiants qui suivent au moins tous les modules avec l'étudiant 'Omar'
4- Donner les noms des enseignants qui n'enseignent pas dans la salle 'X'
5- Donner les noms des étudiants qui sont inscrit à tous les modules
6- Donner les noms des enseignants qui enseignent au moins tous les modules enseigné par l'enseignant 'Bendjima'

Exercice 07

Soit la base de donnée relationnelle suivante :
etudiant(N°etud, nom_etud, adresse)
enseignant(N°ens, nom_ens, grade)
module(N°mod, titre, N°ens)
inscription(N°etud, N°mod, année, appréciation)
Exprimer par l'intermédiaire des opérateurs de l'algèbre relationnelle le résultat des requêtes suivantes :

1- Lister les noms et les adresses des étudiants ayant suivi le module 'BDD'
2- Quels sont les étudiants ayant 'Bendjima' comme enseignant ?
3- donner la liste des numéros des enseignants qui enseignent au moins tous les modules qui sont enseignés par l'enseignant N°09
4- Donner la liste des noms des étudiants qui suivent au moins tous les modules enseignés par 'Bendjima'
5- Donner les numéros des étudiants qui n'ont pas passé d'épreuve en 2022

Exercice 08

salle (nom ,horaire ,titre)
film (titre , realisateur , acteur)
produit (producteur , titre)
vu (spectateur , titre)
aime (spectateur , titre)
Un film est réalisé par un metteur en scène mais peut être financé par plusieurs Producteurs. Un Spectateur peut aimer un film sans l'avoir vu.
Écrire les requêtes suivantes en algèbre relationnel :

1. Dans quelle salle et à quelle heure peut on voir le film ``Mad Max"?
2. Quels sont les Acteurs du film ``Mad Max" ?
3. Quels sont les Acteurs qui ont produit un film ?
4. Quels sont les Producteurs qui regardent les films qu'ils ont produits ?
5. Quels films ne passent à ce moment dans aucune salle ?
6. Quels Spectateurs aiment un film qu'ils n'ont pas vu ?
7. Qui n'aime aucun film qu'il a vu ?
8. Qui a produit un film qui ne passe dans aucune salle ?

Exercice 09

Soit le schéma relationnel :
emp (matr , nom_emp , poste , date_emb , sup , salaire , commission , N°dept)
dept (N°dept , nom_dept , lieu)
projet (code_p , nom_p)
participation (matr , code_p , fonction)

donnez la suite d'opérations élémentaires de l'algèbre relationnelle pour obtenir les données qui correspondent aux listes suivantes :

1. Matricules et des noms des employés qui ont été embauchés avant le 11-09-2022.
2. Noms des employés qui ont le poste de secrétaire.
3. Noms des employés avec le nom du département où ils travaillent.
4. Noms des employés qui travaillent dans le département FINANCES.
5. Numéros de département qui ont des ingénieurs.
6. Matricules des employés qui participent à tous les projets.
7. Noms des employés qui participent à tous les projets.
8. Numéros des départements qui participent à tous les projets.
9. Noms de départements qui ont tous les postes.
10. Noms des employés qui ne participent à aucun projet.
11. Noms des départements qui ont des ingénieurs et des secrétaires.

Exercice 10

Soit la base fabrication suivante :
Client(N°cl, nom, adresse)
Service(N°serv, intitule, localisation)
Piece(N°piece, designation, couleur, poids)
Commande(N°piece, N°serv, N°cl, qte)
Exprimer en Algèbre Relationnel les requêtes suivantes :

1. donner les valeurs de N°serv pour les services qui ont commandé une pièce P1 pour le client C1.
2. donner les valeurs de N°serv pour les services qui ont commandé une pièce de couleur rouge pour le client C1.
3. donner les valeurs de N°cl pour les clients qui ont commander au moins toutes les pièces commandées par S1.
4. donner les valeurs de N°cl pour les clients qui ont commander des pièces figurant uniquement dans les commandes du service S1.

Exercice (supplémentaire)

Soit la base de données stock suivante :
fournisseur(N°fourn, nom_fourn, ville)
produit(N°prod, nom_prod, couleur)
fournir(N°fourn, N°prod, quantite)
Exprimer par l'intermédiaire des opérateurs algébriques les requetes suivantes :

1. Donner les couples de nom des fournisseurs habitant la même ville.
2. Donner les noms des fournisseurs qui fournissent la pièce rouge

Exercice (supplémentaire)

enseignant(nom_ens, age, grade, sal, indice, nom_fac, chef)
etudiant(nom_etud, age, ville)
fac(nom_fac, localité, moyen)
affectation(nom_etud, groupe)
appart_fac(groupe, nom_fac)
Exprimer les requêtes en AR :

1. Trouver les enseignants qui sont des mètres assistants ou des jeunes moins de 30 ans et dont le salaire est égale à 30000DA
2. Trouver les étudiants et les enseignants qui portent le même nom et dont l'âge de l'enseignant ne dépasse pas celui de l'étudiant
3. Trouver le salaire, âge, des maîtres assistant âgé plus de 40 ans
4. Trouver le nom, âge, et la ville des étudiants ne participants pas au groupe 3

Exercice (supplémentaire)

article (N°art, libellé, stock, prix_invent)
fournisseur (N°four, nom_four, adresse_four, ville_four)
acheter (N°four, N°art, prix_achat, delai)
Exprimer les requêtes en AR :

1. Numéros et libellés des articles dont le stock est inférieur à 10
2. Liste des articles dont le prix d'inventaire est compris entre 100 et 300
3. Liste des fournisseurs dont on ne connaît pas l'adresse

Exercice (supplémentaire)

Soit la relation

personne	nom	age	ville
	Brahim	35	Lyon
	Mostefa	29	Béchar
	Amel	20	Naama
	Noura	20	Béchar
	Méliani	24	Lyon
	Soria	20	Oran

Exprimer les requêtes en AR :

1. Les personnes (nom, age, ville) qui habitent Béchar
2. Les personnes (nom, age, ville) qui ont moins de 26ans
3. Les villes dans la relation personne
4. Les noms des personnes habitant à Lyon

Exercice (supplémentaire)

Soit les relations suivantes :
employé (N°emp, nom_emp, profession, date_embauche, salaire, commission, N°depart)
département (N°depart, nom_depart, directeur, ville)
et soit l'exemple suivant :

employé	N°emp	nom_emp	profession	date_embauche	salaire	commission	N°depart
	10	Noura	Docteur	X1	Y1	3000	3
	20	Sara	Ingénieur	X2	Y2	2000	2
	30	Akrem	Homme d'affaire	X3	Y3	5000	1
	40	Amel	Avocate	X4	Y4	5000	3

département	N°depart	nom_depart	directeur	ville
	1	Production	30	Béchar
	2	Développement	20	Alger
	3	Commercial	40	Oran

Exprimer les requêtes suivantes à l'aide de l'algèbre relationnelle, puis les traduire en SQL :
1. Pour obtenir le nom et la profession de l'employé de numéro 10
2. La liste des noms des employés qui travaillent à 'Béchar'
3. Pour avoir le nom du directeur du département 'Commercial'
4. Donner les noms et les salaires des employés
5. Faire le produit cartésien entre les deux relations
6. Donner les noms des employés et les noms de leur département
7. Donner les départements qui n'ont pas d'employés
8. Donner les noms des employés du département 'Commercial' embauchés le même jour qu'un employé du département 'Production'.

Exercice (supplémentaire)

Soit les relations suivantes :
immeuble (adresse_imm_, nbre_etage_imm , date_construction_imm , nom_propriétaire_imm)
appartement (adresse_imm , N°appart , occupant_appart , type_appart , superficie_appart , étage_appart)
personne (nom_per , age_per , profession_per , adresse_per , N°appart)
école (nom_ec , adresse_ec , nbre_classe , nom_directeur)
classe (nom_ec , nom_classe , maitre , nbre_élève_classe)
enfant (nom_pers_resp , prénom_enf , anné_nss , nom_ec , nom_classe)
Exprimer les requêtes suivantes à l'aide de l'algèbre relationnelle, puis les traduire en SQL :
1. Donner l'adresse des immeubles ayant plus de 10 étages et construits avant 1974
2. Donner les noms des personnes qui habitent dans un immeuble dont ils sont propriétaires (occupants et habitants)
3. Donner les noms des personnes qui ne sont pas propriétaires
4. Donner la liste des occupants (nom, age, profession) des immeubles possédés par 'Mostefa'
5. Donner le nom et l'age des maîtres qui habitent dans un immeuble dont le propriétaire est responsable d'un de leurs élèves
6. Donner le nom et l'age des personnes qui sont propriétaires mais qui ne sont ni maître ni directeur d'école.

Exercice (supplémentaire)

Soit la base de données relationnelle suivantes :
Département (N°dep, nom_dep, budget_dep, directeur)
Employé (N°emp, nom_emp, N°dep, N°tel, ville)
Projet (N°proj, titre, budget_proj, N°dep)
Historique (N°proj, N°emp, date, salaire)
Exprimer en langage algébrique les requêtes suivantes :

1. Donner les noms des employés ayant participer au projet 'action' après le 09/09/2001 et dont le salaire était de 42000DA.
2. Lister tous les titres des projets du département informatique.
3. Donner les noms des employés du département informatique participant au projet 'intelligence artificielle'.
4. Donner les noms des employés du département électronique qui n'ont participé à aucun projet.

Exercice (supplémentaire)

Soit le modèle relationnel suivant
Client (N°client , nom , ville)
Commande (N°commande , N°client , date , coût)
Exprimer en algèbre relationnelle les requêtes suivantes :

1. les noms de tous les clients.
2. les numéros des commandes passées à la date 09/09/2001.
3. le numéro du client qui a passé la commande N°09
4. les numéros des commandes et des clients qui ont passé ces commandes à la date 09/09/2022
5. la ville du client qui a passé la commande N°09

Exercice (supplémentaire)

Soient la bases de données suivante :
Avion (N°avion , marque , nbr_places)
Pilote (N°pilote , nom_pil , adresse_pil)
Vol (N°vol , date_vol , N°avion , N°pilote , ville_d , ville_a ,heure_d , heure_a)

Exprimer en AR (schéma et requête) les requêtes suivantes :

- lister les marques de toutes les avions
- lister les noms des pilotes 'Bécharien'
- donnez les marques des avions pilotés par des pilotes 'Bécharien'.

Solutions des exercices sur le langage algébrique

Sol.exo 01

1-

R ⋃ S	X1	X2	X3	X4
	100	a	74	12
	110	b	78	13
	120	c	77	12
	200	d	79	11

R-S	X1	X2	X3	X4
	110	b	78	13
	120	c	77	12

2-

RxS	X1	X2	X3	X4	Y1	Y2	Y3
	110	a	74	12	Mostefa	i	k
	110	a	74	12	Rafik	j	t
	120	b	78	13	Mostefa	i	k
	120	b	78	13	Rafik	j	t

3-

La projection de R sur les attributs X3 et X4

Π (X3,X4)[R]	X3	X4
	74	12
	77	12
	78	13

La restriction de la relation R par la condition X3=77

X1	X2	X3	X4
110	b	77	12
120	c	77	12

La restriction de la relation R par la condition X3=77 et X4>13 est vide

4-

T	X1
	Mostefa
	Omar

Sol.exo 02

R⋃S :

X	**Y**	**Z**
x	y	z
z	y	t
z	k	t
y	w	x
t	x	k

R-S :

X	**Y**	**Z**
x	y	z
z	y	t
z	k	t

RxS :

X	**Y**	**Z**	**U**	**V**	**W**
x	y	z	y	g	x
x	y	z	t	x	k
t	x	k	y	g	x
t	x	k	t	x	k
z	y	t	y	g	x
z	y	t	t	x	k

Project R sur (x,y)

X	**Y**
x	y
t	x
z	y

Selection R pour Y=y

X	**Y**	**Z**
x	y	z
z	y	t

Sol.exo 03

1.

R1 ← SELECT vehicule WHERE type=’R5’
R2 ← NATJOIN R1 WITH possede OVER N°V
R3 ← NATJOIN R2 WITH personne OVER N°SS
R4 ← PROJECT R3 OVER nom

vehicule	N°V	type	……..
	X1	R5	…….
	X2	505	……..
	X3	R5	……..
	X4	405	……..

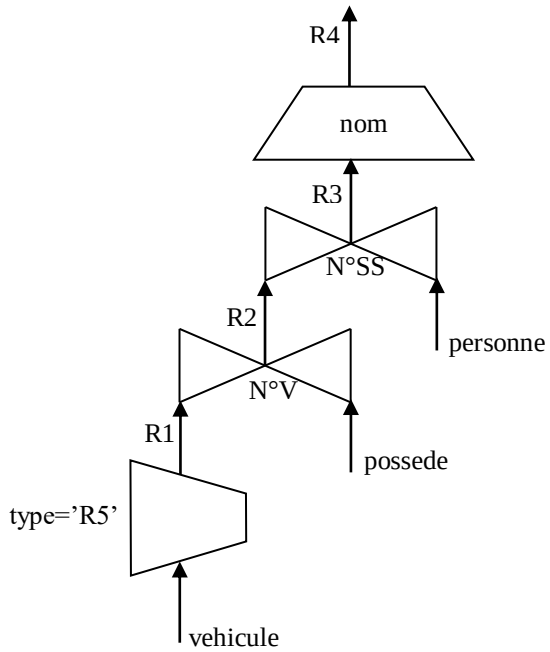
possede	N°SS	N°V	……
	SS1	X1	……
	SS2	X2	…….
	SS3	X3	……..
	SS4	X4	…….

personne	N°SS	nom	……
	SS1	M^{ed}	…….
	SS2	Ali	…….
	SS3	Omar	……..
	SS4	Mostefa	…….

R1	N°V	type	……
	X1	R5	…….
	X3	R5	……..

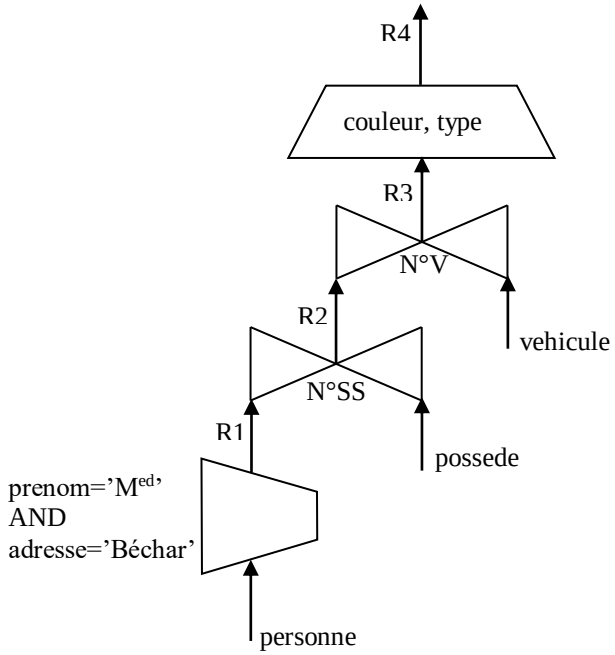
R2	N°V	type	……	N°SS	…….
	X1	R5	……..	SS1	……
	X3	R5	…….	SS3	……

R3	N°V	type	…….	N°SS	…….	Nom	…….
	X1	R5	……	SS1	……	M^{ed}	……
	X3	R5	……	SS3	……	Omar	……



2.

R1 ← SELECT personne WHERE prenom='M^{ed}' AND adresse='Béchar'
R2 ← NATJOIN R1 WITH possede OVER N°SS
R3 ← NATJOIN R2 WITH vehicule OVER N°V
R4 ← PROJECT R3 OVER type, couleur



3.
R1 ← SELECT vehicule WHERE marque=’Renault’
R2 ← NATJOIN R1 WITH possede OVER N°V
R3 ← SELECT R2 WHERE prix<500000DA AND date ≥ 01-01-96 AND date ≤ 10-10-98
R4 ← NATJOIN R3 WITH personne OVER N°SS
R5 ← PROJECT R4 OVER nom

Sol.exo 04

1- R1 ← SELECT stagiaire WHERE ville_stag=’Béchar’
R2 ← PROJECT R1 WITH nom_stag, prenom_stag, origine_stag, ville_stag
2- R1 ← SELECT stage WHERE N°stage=’35’
R2 ← PROJECT R1 WITH date_debut, date_fin
3- R1 ← SELECT stage WHERE N°stage=’10’
R2 ← PROJECT R1 WITH N°prof
R3 ← NATJOIN R2 WITH professeur OVER N°prof
R4 ← PROJECT R3 WITH N°prof, nom_prof, prenom_prof, ville_prof
4- R1 ← SELECT stage WHERE date_debut=’10-01-2001’
R2 ← PROJECT R1 WITH N°salle
R3 ← NATJOIN R2 WITH salle OVER N°salle
R4 ← MINUS salle WITH R3
R5 ← PROJECT R4 WITH N°salle, capacite

Sol.exo 05

1- R1 ← PROJECT representation OVER titre_rep
2- R1 ← SELECT representation WHERE lieu=’Bastille’
R2 ← PROJECT R1 OVER titre_rep
3- R1 ← NATJOIN musicien WITH representation OVER N°rep
R2 ← PROJECT R1 OVER nom, titre_rep
4- R1 ← SELECT programmer WHERE date=09-09-22
R2 ← NATJOIN R1 WITH representation OVER N°rep
R3 ← PROJECT R2 OVER titre_rep, lieu, tarif

Sol.exo 06

1- R1 ← SELECT enseignant WHERE nom=’Bendjima’
R2 ← NATJOIN R1 WITH enseignement OVER N°ens
R3 ← NATJOIN R2 WITH inscription OVER code_mod
R4 ← NATJOIN R3 WITH etudiant OVER N°etud
R5 ← PROJECT R4 OVER nom_etud

2- R1 ← SELECT etudiant WHERE nom_etud=’Omar’
R2 ← NATJOIN R1 WITH inscription OVER N°etud
R3 ← PROJECT R2 OVER code_mod
R4 ← NATJOIN R3 WITH inscription OVER code_mod
R5 ← NATJOIN R4 WITH etudiant OVER N°etud
R6 ← PROJECT R5 OVER nom_etud

3- R1 ← SELECT etudiant WHERE nom_etud=’Omar’
R2 ← NATJOIN R1 WITH inscription OVER N°etud
R3 ← PROJECT R2 OVER code_mod
R4 ← PROJECT inscription OVER code_mod, N°etud
R5 ← DIV R4 BY R3
R6 ← NATJOIN R5 WITH etudiant OVER N°etud
R7 ← PROJECT R6 OVER nom_etud

4. R1 ← SELECT emploi_temps WHERE salle=’X’
R2 ← PROJECT R1 OVER N°ens
R3 ← PROJECT emploi_temps OVER N°ens
R4 ← DIFF(R3, R2)
R5 ← NATJOIN R4 WITH enseignant OVER N°ens
R6 ← PROJECT R5 OVER nom_ens

5. R1 ← PROJECT module OVER code_mod
R2 ← PROJECT inscription OVER code_mod, N°etud
R3 ← DIV R2 BY R1
R4 ← NATJOIN R3 WITH etudiant OVER N°etud
R5 ← PROJECT R4 OVER nom_etud

6. R1 ← SELECT enseignant WHERE nom_ens=’Bendjima’
R2 ← NATJOIN R1 WITH enseignement OVER N°ens
R3 ← PROJECT R2 OVER code_mod
R4 ← PROJECT enseignement OVER code_mod, N°ens
R5 ← DIV R4 BY R3
R6 ← NATJOIN R5 WITH enseignant OVER N°ens
R7 ← PROJECT R6 OVER nom_ens

Sol.exo 07

1- R1 ← SELECT module WHERE titre=’BDD’
R2 ← PROJECT R1 OVER N°mod
R3 ← NATJOIN R2 WITH inscription OVER N°mod
R4 ← NATJOIN R3 WITH etudiant OVER N°etud
R5 ← PROJECT R4 OVER nom_etud, adresse

2- R1 ← SELECT enseignant WHERE nom_ens=’Bendjima’
R2 ← PROJECT R1 OVER N°ens
R3 ← NATJOIN R2 WITH module OVER N°ens
R4 ← PROJECT R3 OVER N°mod
R5 ← NATJOIN R4 WITH inscription OVER N°mod
R6 ← NATJOIN R5 WITH etudiant OVER N°etud
R7 ← PROJECT R6 OVER nom_etud

3- R1 ← SELECT module WHERE N°ens=’09’
R2 ← PROJECT R1 OVER N°mod
R3 ← PROJECT module OVER N°ens, N°mod
R4 ← DIV R3 BY R2

4- R1 ← SELECT enseignant WHERE nom_ens=’Bendjima’
R2 ← PROJECT R1 OVER N°ens
R3 ← NATJOIN R2 WITH module OVER N°ens
R4 ← PROJECT R3 OVER N°mod
R5 ← PROJECT inscription OVER N°etud, N°mod
R6 ← DIV R5 BY R4
R7 ← NATJOIN R6 WITH etudiant OVER N°etud
R8 ← PROJECT R7 OVER nom_etud

5- R1 ← PROJECT inscription OVER N°etud
R2 ← SELECT inscription WHERE année=’2022’
R3 ← PROJECT R2 OVER N°et
R4 ← DIFF R1 WITH R3

Sol.exo 08

1.
R1 ← SELECT salle WHERE titre = 'Mad Max'
R2 ← PROJECT R1 OVER nom , horaire

2.
R1 ← SELECT film WHERE titre = 'Mad Max'
R2 ← PROJECT R1 OVER acteur

3.
R1 ← NATJOIN film WITH produit OVER film.acteur = produit.producteur
R2 ← PROJECT R1 OVER acteur

4.
R1 ← NATJOIN produit WITH vu OVER produit.producteur = vu.spectateur
R2 ← PROJECT R1 OVER producteur

5.
R1 ← PROJECT film OVER titre
R2 ← PROJECT salle OVER titre
R3 ← DIFF R1 WITH R2

6.
R1 ← PROJECT aime OVER spectateur
R2 ← PROJECT vu OVER spectateur
R3 ← DIFF R1 WITH R2

7.
R1 ← NATJOIN vu WITH aime OVER spectateur
R2 ← PROJECT R1 OVER spectateur
R3 ← PROJECT vu OVER spectateur
R4 ← DIFF R3 WITH R2

8.
R1 ← PROJECT film OVER titre
R2 ← PROJECT salle OVER titre
R3 ← DIFF R1 WITH R2
R4 ← NATJOIN R3 WITH produit OVER titre
R5 ← PROJECT R4 OVER producteur

Sol.exo 09

1.
R1 ← SELECT emp WHERE date_emb < 11-09-2001
R2 ← PROJECT R1 OVER matr , nom_emp

2.
R1 ← SELECT emp WHERE poste = ‘secrétaire’
R2 ← PROJECT R1 OVER nom_emp

3.
R1 ← NATJOIN emp WITH dept OVER N°dept
R2 ← PROJECT R1 OVER nom_emp , nom_dept

4.
R1 ← SELECT dept WHERE nom_dept = ‘finance’
R2 ← NATJOIN R1 WITH emp OVER N°dept
R3 ← PROJECT R2 OVER nom_emp

5.
R1 ← SELECT emp WHERE poste = ‘ingenieur’
R2 ← PROJECT R1 OVER N°dept

6.
R1 ← PROJECT participation OVER matr , code_p
R2 ← PROJECT projet OVER code_p
R3 ← DIV R1 with R2

7.
R1 ← PROJECT participation OVER matr , code_p
R2 ← PROJECT projet OVER code_p
R3 ← DIV R1 with R2
R4 ← NATJOIN R3 WITH emp OVER matr
R5 ← PROJECT R4 OVER nom_emp

8.
R1 ← PROJECT participation OVER matr , code_p
R2 ← PROJECT projet OVER code_p
R3 ← DIV R1 with R2
R4 ← NATJOIN R3 WITH emp OVER matr
R5 ← PROJECT R4 OVER N°dept

Sol.exo 10

1. R1 ← select commande where N°p='P1' and N°cl='C1'
   R2 ← project R1 over N°serv

2. R1 ← select piece where couleur='rouge'
   R2 ← select commande where N°cl='C1'
   R3 ← Natjoin R1 with R2 over N°p
   R4 ← project R3 over N°serv

3. R1 ← select commande where N°serv='S1'
   R2 ← project R1 over N°p
   R3 ← project commande over N°cl, N°p
   R4 ← div R3 by R2

4. R1 ← select commande where N°serv='S1'
   R2 ← project R1 over N°cl
   R3 ← project commande over N°cl
   R4 ← diff (R3,R2)

Exercices sur le langage SQL

Exercice 01

Malade (code_mal, nom, prenom, adresse, tel)
Chirurg (code_ch, nom, specialite)
Occupe (code_mal, date_deb, date_fin, num_lit)
Opere (code_ch, code_mal, date_op, nature, heure)

1- Donner le code des chirurgiens qui sont travaillé le 09-09-22
2- Donner le nom et le prénom des personnes qui sont rentrée à l'hôpital le 05-03-22
3- Donner le nombre d'intervention de chaque chirurgien entre le 09-05-22 et 09-09-22
4- Le nom des chirurgiens ayant travaillé au moins une fois à la même date que le chirurgien 'M^{ed}'

Exercice 02

Soit la base de données relationnelle suivante :
client(N°cl, nom_cl, adresse_cl)
article(N°art, nom_art, prix_achat_art, prix_vente_art, poids_art, coul_art)
commande(N°cde, N°cl, date_cde, N°four)
ligne_cde(N°cde, N°ligne, N°art, qte_cde)
fournisseur(N°four, nom_four, adresse_four)

Exprimer les requêtes suivantes en SQL :
1- Donner tous les articles pour lesquels le prix de vente est supérieur ou égal au double de prix d'achat
2- Donner tous les articles rouges de poids >100g
3- Pour obtenir les articles qui ne sont pas rouges et dont le poids est inférieur ou égal à 500g
4- Pour rechercher les fournisseurs qui pendant la période du 09-05-22 au 09-09-22 ont réalisé plus de 10 vente
5- Pour calculer le prix de vente moyen des articles de chaque couleur en excluant les articles pour lesquels le prix d'achat est inférieur à 500DA
6- Pour rechercher tous les articles dont le poids est inférieur au poids de l'article numéro'A02'
7- Pour rechercher dans la table articles, les articles de même couleur que l'article 'A10' et dont le poids est supérieur ou égal au poids moyen de tous les articles
8- On veut rechercher la liste des articles dont le prix de vente est supérieur au prix de vente de l'article de couleur blanche de moins chère

Exercice 03

Soit la base de données suivante :
Boisson (N°bois, origine, date_fab, concentration)
Buveur (N°buv, nom, adresse)
Bus (N°buv, N°bois, quantité)

Exprimer en SQL :

1. Donner les origines et les numéros de la boisson fabriquée en 09-09-2022, et concentration supérieur à 10 triés par ordre alphabétique croissant sur origine et décroissant sur numéro de boisson
2. Donner les noms des buveurs ayant bus un boisson d'Abricot
3. Donner les numéros de boisson bus par plus de 100 buveurs
4. Insérer le tuple (200, orange, 09-09-2022, 14) dans la relation boisson
5. Supprimer tous les boissons bus par 'Bendjima'
6. Mettre à jour les quantités bus à 0 pour tous les buveurs d'adresse Béchar

Exercice 04

Soit la base de données relationnelle suivante :
Transport(client, gare_origine, gare_destination, N°wagon, type_marchandise, poids_marchandise, date_chargement)
Réseau (gare_origine, gare_destination, gare_suivante, N°ligne)
Ligne (N°ligne, rang, gare)
Trafic (N°train, N°ligne, jour)
Train (N°train, N°wagon)
Wagon (N°wagon, type_wagon, poids_vide, capacite, etat, gare)

Exprimer les requêtes suivantes en SQL :

1. Donner la liste des numéros de wagon de type 'frigo' disponible à la gare de tours
2. Donner la liste des types des wagons du train 4002
3. Construire une relation qui donne pour le train 4002 et pour chaque wagon, son numéro, le poids des marchandises et le poids à vide
4. Donner les numéros de lignes tels qu'il existe un train tous les jours
5. Donner les numéros de lignes tels qu'il existe un train lundi et jeudi
6. Construire une relation qui donne pour chaque numéro de train le nombre de wagons
7. donner les numéros de train ayant plus de 100 wagons
8. Quelle est la gare d'arrivé de la ligne 35

Exercice 05

Soit la base de données de l'université :
etudiant(nom_etud, age_etud, ville)
enseignant(nom_ens, age_ens, nom_fac, grade, salaire)
faculte(nom_fac, localite, recteur)
affectation(nom_et, groupe)
emploi_temps(nom_ens, groupe, matiere, salle, jour, heure, activite_pedag)

1- déterminer les clés primaires et étrangères de ces relations
2- créer en SQL les relations : emploi_temps, faculte
3- exprimer en SQL les requêtes suivantes :
   Q1 :trouver le couple des noms différents des étudiants habitant la même ville.
   Q2 :trouver toutes les informations concernant les étudiants n'habitant pas la ville Béchar
   Q3 :trouver le nom et l'âge des étudiants affectés au groupe 3 et dont le nom se termine par la lettre 'B'
   Q4 :trouver le nom et le salaire des enseignants dont le salaire dépasse moyenne des salaires et dont l'âge est compris entre 35 et 50 ans
   Q5 :trouver les noms des enseignants de moins de 35 ans, de grade 'chargé de cours' qui enseignent la matière 'BDD' à la fac d'Oran dont le recteur est M^{ed} .

Exercice 06

client(id_cl, nom, adresse, code_post, ville, tel)
article(id_art, designation, prix_unit, qte_stock)
commande(N°cde, id_cl, id_vend, date_cde)
ligne_cde(N°cde, N°ligne, id_art, qte_cde)
vendeur(id_vend, nom_vend, quantite, salaire)

1- Pour visualiser tout les données concernant les clients
2- On désire connaître seulement le nom et le numéro de téléphone des clients
3- Pour visualiser uniquement les clients qui habitent à Béchar
4- Lister tous les articles dont le prix unitaire est > à 150DA et dont quantité en stock est ≤ à 100 et toutes les commandes enregistrés après la date 09-09-99
5- Lister tous les articles dont le prix est compris entre 150 et 200 ainsi que toutes les commandes non enregistrer entre la date 09-09-99 et 30-01-00
6- Lister tous les clients des villes 'Béchar' 'Oran' 'Alger' ainsi que tous les articles dont le prix unitaire est 150, 200 ou 300

Exercice 07

hôtels (nom_hot, ville_hot, lits_dip, date)
vols (N°vol, compagnie, ville_dep, ville_arr, date_dep, date_arr, places_disp)
voyages (N°voyage, N°vol_all, N°vol_ret, nom_hot)

Exprimer en SQL les requêtes suivantes :
1. Les numéros de voyage et le et le nombre de places disponibles dans les vols aller et retour pour chaque voyage.
2. Le nombre d'hôtels à Béchar.
3. le nombre de lits disponibles à Béchar le 09-09-2002
4. Le nombre de lits disponible pour le voyage 113 (les lits sont limitées par le nombre de lits disponibles dans l'hôtel pendant la période du voyage).
5. Toutes les villes dans la table hôtels et pour chaque ville le nombre d'hôtels.

Exercice 08

Soit les trois relation suivantes :

```
CREATE TABLE fournisseur
(N°four  char(5) not null,
nom_four char(20) , status integer,
ville char(10) ,
PRIMARY KEY(N°four)) ;

CREATE TABLE produit
(N°prod  char(6) not null,
nom_prod char(20) ,  couleur char(6),
poids integer,
PRIMARY KEY(N°prod)) ;

CREATE TABLE client
(N°cl  char(6) not null,
nom_cl char(20) ,
ville char(10) ,
PRIMARY KEY(N°cl)) ;

CREATE TABLE commande
(N°four  char(5) not null,
N°prod  char(6) not null,
N°cl  char(6) not null,
Qte integer,
PRIMARY KEY(N°four , N°prod , N°cl)) ;
```

Exprimer les requêtes suivantes en SQL:

1. Toutes les informations sur les clients.
2. Toutes les informations sur les clients à 'Béchar'.
3. La liste triée des numéros des fournisseurs du client avec le numéro C1.
4. Les commandes avec une quantité entre 300 et 750.
5. Les commandes avec une quantité différente de NULL.
6. Les numéros des clients qui sont situés dans une ville qui commence par "B".
7. Les numéros des fournisseurs et des clients qui sont situés dans la même ville.
8. Les numéros des fournisseurs et des clients qui ne sont pas situés dans la même ville.
9. Les numéros des produits fournis par des fournisseurs à Béchar.
10. Les numéros des produits fournis par des fournisseurs 'Béchar' à des clients 'Oran'.
11. Les couples de villes (v_i , v_j) tel qu'il existe au moins un fournisseur dans la ville v_i d'un client dans la ville v_j .
12. Les numéros des produits fournis à des clients situés dans la même ville que leurs fournisseurs.

Exercice 09

Soit le modèle relationnel suivant relatif à la gestion des notes annuelles d'une promotion d'étudiants :
etudiant (N°etud, nom_etud, prénom_etud)
matière (code_mat, libellé_mat, coeff_mat)
evaluer (N°etud, code_mat, date, note)

Exprimer les requêtes suivantes en SQL :

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 10

Soient la bases de données suivante :
Avion (N°avion , marque , nbr_places)
Pilote (N°pilote , nom_pil , adresse_pil)
Vol (N°vol , date_vol , N°avion , N°pilote , ville_d , ville_a ,heure_d , heure_a)

Exprimer en **SQL** les requêtes suivantes :

1. Donnez le nombre de pilotes 'Bécharien'
2. Donnez le nombre de places pour tous les avions
3. Donnez tous les vols effectués par des pilotes 'Bécharien'
4. Donnez les noms des pilotes qui ont pilotés tous les avions.

Exercice (supplémentaire)

Soit la relation suivante :
transport(N°train, N°wagon, type_wagon, poids_vide, capacité, état, gare, gare_destination, N°ligne, rang_ligne, date)
Soit les dépendances fonctionnelles les suivantes :
N°wagon $\rightarrow$capacité ; N°wagon $\rightarrow$type_wagon ; N°wagon $\rightarrow$poids_vide ;
N°wagon $\rightarrow$etat ; N°wagon $\rightarrow$gare_destination
N°ligne $\rightarrow$gare ; N°ligne $\rightarrow$rang_ligne ;
N°train $\rightarrow$N°ligne ; N°train $\rightarrow$date

- La relation transport est elle en 3FN ? sinon décomposée la ?
- Exprimer en SQL :
  1. Donner la liste des lignes qui partent de la gare d'Alger rangé par ordre croissant
  2. Donner pour chaque wagon sa capacité, et son poids vide
  3. Donner pour chaque wagon du train 2300, son numéro, sa capacité, et son poids vide
  4. Donner la liste des numéros de wagon qui partent d'Oran le 09-09-22 et leurs capacité
  5. Donner les numéros de wagons chargés du train 4002 ainsi que leur type

Exercice (supplémentaire)

Soit la base de données suivantes
buveur(code_buv, nom_buv, ville)
boisson(code_boiss, nom_boiss, couleur, degré, année)
bus(code_buv, code_boiss, quantité)
producteur(code_prod, nom_prod, région)
produire(code_prod, code_boiss, quantite)

Créer en SQL les relations :boisson, bus
Exprimer en SQL les requêtes suivantes :

1. Donner les noms de boisson sans double
2. Donner les buveurs ayant bu de boisson dont le nom commence par 'B', degré inconnu
3. Donner les noms des boissons bus par au moins un buveur
4. Calculer la moyenne des degrés pour chaque boisson
5. Insérer les producteurs de la région de Béchar ayant produit des boissons de 2022 dans la relation buveur

Exercice (supplémentaire)

enseignant(nom_ens, age, grade, sal, indice, nom_fac, chef)
etudiant(nom_etud, age, ville)
fac(nom_fac, localité, moyen)
affectation(nom_etud, groupe)
appart_fac(groupe, nom_fac)

Exprimer les requêtes suivantes en SQL :
1. Trouver le nom de la fac a qui appartient l'étudiant 'Omar'
2. Trouver les salaires de tous les enseignants
3. Trouver le salaire et l'âge pour l'enseignant M^{ed} âgé de 40 ans

Exercice (supplémentaire)

livre (code_livre, titre_livre, editeur, année_edition, prix_achat, date_achat)
auteur (N°auteur, nom_auteur, prénom_auteur)
cassette (N°cassette, état, durée_enreg, durée_disp, prix_achat, date_achat)
film (N°film, titre_film, categorie, N°cassette)
emprunteur (N°emp, nom_emp, prénom_emp, adresse_emp)
ecrire (N°auteur, code_livre)
emprunter_livre (N°emp, code_livre, date_emp, date_retour)
emprunter_film (N°emp, N°film, date_emp, date_retour)

Ecrire en SQL, les requêtes permettant d'obtenir les réponses aux questions suivantes:
1. quelle sont les titres et catégories des films enregistrés sur la cassette numéro 3 ? les titres seront rangés par ordre alphabétique
2. quels sont les livres qui ont été empruntés et combien de fois l'on-ils été ? en résultat :les titres et le nombre d'emprunts rangés par ordre décroissant du nombre d'emprunts et pour un même nombre, par ordre alphabétique des titres
3. quels films ont été empruntés plusieurs fois par une même personne ? le résultat fournira le numéro d'emprunteur, son nom, son prénom et le titre du film
4. quel est le prix moyen des cassettes préenregistrées ?
5. quelles sont les cassettes disponibles ? en résultat on donnera les numéros des cassettes rangés dans l'ordre croissant et pour chacun la liste des titres des films qu'elle contient (rangés par ordre alphabétique). Le numéro des cassettes ne sera pas répété.

Exercice (supplémentaire)

Soit la base de données relationnelle suivantes :
Département (N°dep, nom_dep, budget_dep, directeur)
Employé (N°emp, nom_emp, N°dep, N°tel, ville)
Projet (N°proj, titre, budget_proj, N°dep)
Historique (N°proj, N°emp, date, salaire)

1. Déterminer les clés primaires et les clés étrangères de ces relations.
2. crées en **SQL** les relations PROJET et HISTORIQUE.
3. Exprimer en **SQL** les requêtes suivantes :
   - donner les numéros des employés habitant dans la même ville que l'employé N°113
   - donner les villes des employés ayant participer au projet 'Bases de données'
   - donner le budget le plus élevé des projets du département N°09
   - supprimer le projet 'intelligence artificielle'.

Exercice (supplémentaire)

Etudiant

nom	ville	age
Reda	Alger	18
Omar	Alger	19
Méliani	Béchar	20

Stagiaire

nom	ville	age
Dalila	Alger	21
Fatiha	Alger	17
Omar	Béchar	20

Donner les résultats des requêtes **SQL** suivantes :

- SELECT nom FROM etudiant
- SELECT age FROM etudiant WHERE ville = 'Béchar'
- SELECT nom , ville FROM stagiaire WHERE age>21
- SELECT nom FROM stagiaire
  INTERSECT
  SELECT nom FROM etudiant
- SELECT count(*) FROM etudiant

Solutions des exercices sur le langage SQL

Sol. Exercice 01

1- SELECT code_ch
FROM opere
WHERE date_op='09-09-22'

2- SELECT nom, prenom
FROM malade
WHERE code_mal IN (SELECT code_mal
FROM occupe
WHERE date_deb='05-03-22')

3- SELECT code_ch, COUNT(*)
FROM chirurg, opere
WHERE chirurg.code_ch=opere.code_ch
AND date_op BETWEEN 09-05-22 AND 09-09-22
GROUP BY code_ch

4- SELECT nom
FROM opere, chirurg
WHERE chirurg.code_ch=opere.code_ch
AND date_op IN (SELECT date_op
FROM chirurg, opere
WHERE chirurg.nom='M^{ed}'
AND chirurg.code_ch=opere.code_ch)

Sol. Exercice 02

1- SELECT *
FROM article
WHERE prix_vente_art >= 2*prix_achat_art

2- SELECT *
FROM article
WHERE coul_art = 'rouge'
AND poids_art > 100

3- SELECT *
FROM article
WHERE coul_art $\neq$ rouge AND pods_art $\leq$ 500

4- SELECT N°four
FROM commande
WHERE 09-05-22 $\leq$ date_cde $\leq$ 09-09-22
GROUP BY N°four
HAVING COUNT(*) > 10

5- SELECT coul_art, AVG(prix_vente_art)
FROM article
WHERE prix_achat_art ≥ 500
GROUP BY coul_art

6- SELECT *
FROM article
WHERE poids_art < (SELECT poids_art
FROM article
WHERE N°art='A02')

7- SELECT N°art, nom_art
FROM article
WHERE coul_art = (SELECT coul_art
FROM article
WHERE N°art='A10')
AND poids_art ≥ (SELECT AVG(poids_art)
FROM article)

8- SELECT nom_art
FROM article
WHERE prix_vente_art>(SELECT MIN(prix_vente_art)
FROM article
WHERE coul_art='blanche'

Sol. Exercice 03

1. SELECT origine, N°bois
FROM boisson
WHERE date_fab='09-09-2022'
AND concentration>10
ORDER BY origine ASC, N°bois DESC

2. SELECT nom
FROM buveur, bus
WHERE buveur.N°buv=bus.N°buv
AND bus.N°bois=boisson.N°bois
AND boisson.origine='Abricot'

3. SELECT N°bois
FROM bus
GROUP BY N°bois HAVING COUNT(*)>100

4. INSERT INTO boisson<200, orange, 09-09-2022, 14>

5. DELETE boisson
WHERE 'Bendjima' IN (SELECT nom
FROM buveur, bus
WHERE boisson.N°bois=bus.N°bois
AND bus.N°buv=buveur.N°buv)

6. UPDATE bus
SET quantité=0
WHERE 'Béchar' IN (SELECT adresse
FROM buveur
WHERE bus.N°buv=buveur.N°buv)

Sol. Exercice 04

1- SELECT N°wagon
FROM wagon
WHERE type_wagon='frigo'
AND etat='libre'
AND gare='tours'

2- SELECT type_wagon
FROM wagon
WHERE N°wagon IN (SELECT N°wagon
FROM train
WHERE N°train='4002'

3- SELECT N°wagon, poids_vide, poids_marchandise
FROM train, wagon, transport
WHERE train.N°wagon = wagon.N°wagon
AND train.N°wagon = transport.N°wagon
AND train.N°train = '4002'

4- SELECT N°ligne
FROM trafic
GROUP BY N°ligne HAVING SET(jour) CONTAIN (SELECT jour
FROM trafic)

5- SELECT N°ligne
FROM trafic
GROUP BY N°ligne
HAVING SET(jour) CONTAIN ('lundi', 'jeudi')

Autre méthode :

SELECT N°ligne
FROM trafic
WHERE jour='lundi'
INTERSECT
SELECT N°ligne
FROM trafic
WHERE jour='jeudi'

6- SELECT N°train, COUNT(N°wagon)
FROM train
GROUP BY N°train

7- SELECT N°train
FROM train
GROUP BY N°train
HAVING COUNT(N°wagon) ≥ 100

8- SELECT gare
FROM ligne
WHERE rang = (SELECT MAX(rang)
FROM ligne
WHERE N°ligne='35')
AND N°ligne = '35'

Sol. Exercice 05

1- clés primaires :
etudiant ⇒ nom_etud
enseignant ⇒ nom_ens
faculte ⇒ nom_fac
affectation ⇒ nom_etud, groupe
emploi_temps ⇒ nom_ens, jour, heure
clés étrangères :
enseignant ⇒ nom_fac % à faculte
affectation ⇒ nom_etud % à etudiant
emploi_temps ⇒ nom_ens % à enseignant

2- création des tables en SQL :

```
CREATE TABLE faculte
(nom_fac char(20) not null,
localite char(20), recteur char(15))
PRIMARY KEY nom_fac ;

CREATE TABLE emploi_temps
(nom_ens char(15) not null,
groupe integer, matière char(10),
jour char(8) not null, heure char(15) not null,
activite_pedag char(15),
PRIMARY KEY(nom_ens, jour, heure),
FORCEGN KEY nom_ens, IDENTIFIES enseignant ;
```

3- Exprimer en SQL :

R1 :SELECT A.nom_etud, B.nom_etud
FROM etudiant A, etudiant B
WHERE A.ville = B.ville
AND A.nom_etud <> B.nom_etud

R2 :SELECT *
FROM etudiant
WHERE ville <> 'Béchar'

Autre méthode :

```
SELECT *
FROM etudiant
WHERE ville not IN (SELECT ville
                    FROM etudiant
                    WHERE ville='Béchar'
```

R3 :SELECT nom_etud, age_etud
FROM etudiant
WHERE nom_etud LIKE '%B'
AND nom_etud IN (SELECT nom_etud
FROM affectation
WHERE groupe=4)

Autre méthode :

```
SELECT nom_etud, age_etud
FROM etudiant, affectation
WHERE nom_etud LIKE '%B'
AND groupe=4
AND etudiant.nom_etud=affectation.nom_etud
```

R4 :SELECT nom_ens, salaire
FROM enseignant
WHERE age_ens BETWEEN 35 AND 50
AND salaire>(SELECT AVG(salaire)
FROM enseignant)

R5 :SELECT nom_ens
FROM emploi_temps
WHERE matiere='BDD'
AND nom_ens IN (SELECT nom_ens
FROM enseignant
WHERE grade = 'charge de cours'
AND age_ens<35
AND nom_fac IN (SELECT nom_fac
FROM faculte
WHERE localite='ORAN'
AND recteur='M^{ed}'

Sol. Exercice 06

1.
```
SELECT *
FROM client
```

2.
```
SELECT nom, tel
FROM client
```

3.
```
SELECT *
FROM client
WHERE ville=’Béchar’
```

4.
```
SELECT *
FROM article
WHERE prix_unit>150DA
AND qte_stock≤100
UNION
SELECT *
FROM commande
WHERE date_cde > 09-09-99
```

5.
```
SELECT *
FROM article
WHERE prix_unit≥150
AND prix_unit≤200
UNION
SELECT *
FROM commande
WHERE date_cde BETWEEN 09-09-99 AND 30-01-00
```

6.
```
SELECT *
FROM client
WHERE ville=’Béchar’ OR ville=’Oran’ OR ville=’Alger’
UNION
SELECT *
FROM article
WHERE prix_unit=150
OR prix_unit=200
OR prix_unit=300
```

Sol. Exercice 07

1. SELECT N°voyage , A.places_disp , B.places_disp
   FROM vol A , vol B , voyages
   WHERE A.N°vol = N°vol_all
   AND B.N°vol = N°vol_ret

2. SELECT COUNT(nom_hot)
   FROM hotels
   WHERE ville_hot = ‘Béchar’

3. SELECT SUM(lits_disp)
   FROM hotels
   WHERE ville_hot = ‘Béchar’
   AND date = ’09-09-2002’

4. SELECT MIN(lits_disp)
   FROM hotels , voyages
   WHERE hotels.nom_hot = voyages.nom_hot
   AND voyages.N°voyage = ‘113’
   AND hotels.date >= voyages.date_dep
   AND hotels.date <= voyages.date

5. SELECT ville_hot , COUNT(nom_hot)
   FROM hotels
   GROUP BY ville_hot

Sol. Exercice 08

1.
SELECT N°cl, nom_cl, ville
FROM client
ou
SELECT *
FROM client

2.
SELECT *
FROM client
WHERE ville = ’Béchar’

3.
SELECT DISTINCT N°four
FROM command
WHERE N°cl = ’C1’
ORDER BY N°four

4.
```
SELECT *
FROM commande
WHERE Qte >= 300 AND Qte <= 750
```
ou
```
SELECT *
FROM commande
WHERE Qte BETWEEN 300 AND 750
```

5.
```
SELECT *
FROM commande
WHERE Qte IS NOT NULL
```
ou
```
SELECT *
FROM commande
WHERE Qte = Qte
```

6.
```
SELECT N°cl
FROM client
WHERE ville LIKE 'B%'
```
ou
```
SELECT N°cl
FROM client
WHERE SUBSTR (ville, 1, 1) = 'B'
```

7.
```
SELECT N°four, N°cl
FROM fournisseur, client
WHERE fournisseur.ville = client.ville
```

8.
```
SELECT N°four, N°cl
FROM fournisseur, client
WHERE NOT fournisseur.ville = client.ville
```
ou
```
SELECT N°four, N°cl
FROM fournisseur, client
WHERE fournisseur.ville <> client.ville
```

9.
```
SELECT DISTINCT N°prod
FROM commande, fournisseur
WHERE commande.N°four = fournisseur.N°four
AND ville = 'Béchar'
```

10.
```
SELECT DISTINCT N°prod
FROM commande, fournisseur, client
WHERE commande.N°four = fournisseur.N°four
AND commande.N°cl = client.N°cl
AND fournisseur.ville = 'Béchar'
AND client.ville = 'Oran'
```

11.
```
SELECT DISTINCT fournisseur.ville, client.ville
FROM commande, fournisseur, client
WHERE commande.N°four = fournisseur.N°four
AND commande.N°cl = client.N°cl
```

12.
```
SELECT DISTINCT N°prod
FROM commande, fournisseur, client
WHERE commande.N°four = fournisseur.N°four
AND commande.N°cl = client.N°cl
AND client.ville = fournisseur.ville
```

Sol. Exercice 09

1.
```
SELECT count(N°etud)
FROM etudiant
```

2.
```
SELECT max(note),min(note)
FROM evaluer
```

3.
```
SELECT libelle_mat, nom_etud, avg(coeff_mat*note)
FROM etudiant, matiere, evaluer
WHERE etudiant.N°etud=evaluer.N°etud
AND matiere.code_mat=evaluer.code_mat
GROUP BY libelle_mat, nom_etud
```

4.
```
SELECT code_mat, avg(coeff_mat*note)
FROM matiere, evaluer
WHERE matiere.code_mat=evaluer.code_mat
GROUP BY code_mat
```

5.
```
SELECT N°etud , avg(coeff_mat*note)
FROM etudiant , matiere , evaluer
WHERE etudiant.N°etud=evaluer.N°etud
AND matiere.code_mat=evaluer.code_mat
GROUP BY N°etud
```

6.
```
SELECT avg(coeff_mat*note)
FROM matiere , evaluer
WHERE matiere.code_mat = evaluer.code_mat
```

7. SELECT nom_etud , avg(coeff_mat*note)
   FROM etudiant , matiere , evaluer
   WHERE etudiant.N°etud=evaluer.N°etud
   AND matiere.code_mat=evaluer.code_mat
   GROUP BY nom_etud
   HAVING avg(coeff_mat*note)>(SELECT avg(coeff_mat*note)
   FROM matiere, evaluer
   WHERE matiere.code_mat=evaluer.code_mat)

Sol. Exercice 10

1. SELECT count(N°pilote)
   FROM pilote
   WHERE adresse_pil=’Béchar’

2. SELECT SUM(nbre_places)
   FROM avion

3. SELECT N°vol
   FROM vol
   WHERE N°pilote IN (SELECT N°pilote
   FROM pilote
   WHERE adresse_pil=’Béchar’)

4. SELECT nom_pil
   FROM pilote, vol
   WHERE pilote.N°pilote=vol.N°pilote
   AND N°avion=all(SELECT N°avion
   FROM avion)

Bibliographie

[1] : Kimball R., Ross M., Thornthwaite W., Mundy J., Becker B. The Data Warehouse Lifecycle Toolkit. Wiley Publishing, second edition. 2008.

[2] : Tardieu H., Rochfeld A., Colleti R., Méthode MERISE Tome 1: Principes et outils, Les Editions d'Organisation, 1983.

[3] : Arribe Thibaut. Conception des chaînes éditoriales: l'activité et structurer le graphe documentaire pour améliorer la maîtrise de la rééditorialisation. Université de Technologie de Compiègne, Mémoire de Doctorat. 2014.

[4] : Codd EF, A relational model for large shared data banks, Communications de l'ACM, juin 1970.

[5] : Tardieu H., Rochfeld A., Colleti R., Panet G., Vahee G., Méthode MERISE Tome 2: Démarche et pratiques, Les Editions d'Organisation, 1985.

[6] : Philippe Guézélou. Modélisation des données : Approche pour la conception des bases des données. http://philippe.guezelou.free.fr/mcd/mcd.htm, 2006.

[7] : Foucault Odile, Thiéry Odile, Kamel Smaïli. Conception des systèmes d'information et programmation événementielle: de l'étape conceptuelle à l'étape d'implantation. Intereditions, 1996.

[8] : Pierre Crescenzo. Un support de cours magistraux de Bases de données. http://www.crescenzo.nom.fr/CMBasesDeDonnees, 2005.

[9] : Gardarin Georges. Bases de données : objet et relationnel. Eyrolles, 1999.

[10] : Sikha Bagui et Richard Earp, Database Design Using Entity-Relationship Diagrams, CRC Press. 2011.

[11] : Bourda Yolaine, Bases de Données Relationnelles et Systèmes de Gestion de Bases de Données Relationnels, le Langage SQL, 2009.

[12] : Pascal Roques, UML 2 par la pratique, 7e édition, Eyrolles, 2009.

[13] : Michelle Clouse, Algèbre relationnelle Guide pratique de conception d'une base de données relationnelle normalisée. Édition : ENI - 377 pages , 2008.

[14] : Thibaud Cyril. MySQL 4: Installation, mise en oeuvre et programmation. Editions ENI, Collection Ressources Informatiques.2003.

[15] : Jérôme Gabillaud. SQL et algèbre relationnelle - Notions de base. TechNote. ENI, February 2004.

[16] : Articles en ligne sur Developpez.com. LE SQL de A à Z : Les jointures, ou comment interroger plusieurs tables. https://sql.developpez.com/sqlaz/jointures, 2005.

[17] : Articles en ligne sur Developpez.com. LE SQL de A à Z : Le simple (?) SELECT et les fonctions SQL. https://sql.developpez.com/sqlaz/select, 2005.

[18] : Cyril Gruau. Conception d'une base de données. https:// cyril-gruau.developpez.com/uml/tutoriel/ConceptionBD, 2006.

[19] : Yolaine Bourda. Systèmes de Gestion de Bases de Données Relationnelles. http://wwwlsi.supelec.fr/www/yb/poly_bd/poly.html, 2005.

[20] : Chris-J Date, Introduction aux bases de données. Vuibert - 1045 pages, 2004.

[21] : Jacky Akoka and Isabelle Comyn-Wattiau. Conception des bases de données relationnelles. Vuibert informatique, 2001.

[22] : Colin Ritchie, Database Principles and Design, Cengage Learning EMEA - 2008.

[23] : Borderie Xavier, Cinq petits conseils pour un schéma UML efficace, http://developpeur.journaldunet.com/tutoriel/cpt/031013cpt_uml5conseils.shtml, 2004.

[24] : SQL Anywhere Studio. ASA : Programmation avec Embedded SQL. http://www.ianywhere.com/developer/product_manuals/sqlanywhere/0901/fr/html , 2005.

[25] : Anne-Christine Bisson, SQL - Les fondamentaux du langage. Editions ENI; 3e édition, 409 pages. 13 décembre 2017.

[26] : Bruchez Rudi. Les bases de données NoSQL et le BigData: Comprendre et mettre en oeuvre. 2ème édition, Eyrolles. 2015.

[27] : Philip J. Pratt et Joseph J. Adamski, Concepts of Database Management, Cengage Learning - 2011.

[28] : The PostgreSQL Global Development Group. Documentation PostgreSQL 7.4.7. http://traduc.postgresqlfr.org/pgsql-fr/index.html, 2005.

[29] : Jean-Luc Hainaut, Bases de données. Concepts, utilisation et développement. http://www.developpez.com/hcesbronlavau/Transactions.htm, 2002.

[30] : Kimball R., Ross M. Entrepôts de données: guide pratique de modélisation dimensionnelle. Vuibert. 2003.

[31] : Goglin J.-F. La construction du datawarehouse: du datamart au dataweb. Hermes, 2ème édition. 2001.

[32] : Mata-Toledo Ramon A., Cushman Pauline K.. Programmation SQL. Ediscience, 2003.