Bases de données médicales : Ce polycopié s'adresse aux étudiants de la troisième année licence de la filière nationale en Génie Biomédical. IL concerne la matière intitulée : base…

Book · October 2018

CITATIONS
0

READS
906

1 author:

Abderrahim Mohammed El Amine
Université Kasdi Merbah Ouargla
29 PUBLICATIONS **128** CITATIONS

SEE PROFILE

الجـــمـــهـــوريـــة الجـــزائـــريـــة الديمـــقـــراطيـــة الـشـــــعـبـيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــعـــلـيـــم العـــالي و البـــحث العـــلـــمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــامعة أبي بكـــر بـلـقـايد – تـلمســـــان –

Université Aboubakr Belkaïd – Tlemcen –

Polycopié de cours et travaux dirigés

Pour les étudiants de Licence
Option : Informatique Biomédicale de la filière nationale Génie Biomédical

Intitulé : Bases de données médicales

ABDERRAHIM Mohammed El Amine

Etablissement de rattachement : Université de Tlemcen
Faculté : Technologie
Département : Génie Biomédical

ANNEE UNIVERSITAIRE 2017/ 2018

Descriptif du polycopié

Ce polycopié s'adresse aux étudiants de la troisième année licence de la filière nationale en Génie Biomédical. IL concerne la matière intitulée : bases de données médicales (GB 618). Il a pour objectif l'étude des bases de données et particulièrement les bases de données relationnelles.

Ce polycopié est structuré en trois chapitres.

Le premier chapitre introduit les concepts fondamentaux des bases de données.

Le deuxième chapitre aborde le modèle relationnel qui est considéré comme le plus simple des modèles de bases de données due à sa vision tabulaire des données très intuitive et aux nombreux avantages dus au fait qu'il soit basé sur la théorie des ensembles : Langage de manipulation des données ensemblistes grâce à l'algèbre relationnelle et grâce à des langages qui permettent de spécifier ce que l'on souhaite obtenir sans dire comment l'obtenir.

Le troisième chapitre est consacré à l'étude du langage SQL (Structured Query Language) qui peut être considéré comme le langage d'accès normalisé aux bases de données. Il est aujourd'hui supporté par la plupart des produits commerciaux que ce soit par les systèmes de gestion de bases de données micro tel que « Access » ou par les produits plus professionnels tels que « Oracle ». Il a fait l'objet de plusieurs **normes ANSI/ISO** dont la plus répandue aujourd'hui est la norme SQL2 qui a été définie en 1992. Ce chapitre décrit les principaux aspects de cette norme.

Vers la fin du chapitre deux et trois, des exercices avec solutions sont proposés.

Objectifs de l'enseignement

- Comprendre le fonctionnement des SGBD.
- Construire des schémas de relation.
- Réaliser les opérations de l'algèbre relationnelle.
- Créer, mettre à jour et interroger une base de données en SQL.

Table des matières

Chapitre 3: Le langage SQL

Chapitre 1

Bases de données, concepts fondamentaux

1.1 Bases de données, banques de données et fichiers

1.1.1 Base de Données (BD)

Une **BD** représente l'ensemble (cohérent, intégré, partagé) des informations nécessaires au fonctionnement d'une entreprise[1]. La gestion de ces informations est assurée par un logiciel appelé **Système de Gestion de Bases de Données** (SGBD).

[1] Une entreprise : Toute collectivité d'individus travaillant en coordination à la réalisation d'un objectif commun. Exemple d'entreprise: un institut de formation.

Exemples de BD : Celle qui permet la gestion des personnels, étudiants, cours, et inscriptions d'une université ou école, celle qui permet la gestion du dossier patient d'un hôpital, celle du système de réservation de places d'avion des compagnies d'aviation, celle qui permet la gestion des comptes des clients des sociétés bancaires, etc.

1.1.2 Banque de données

Une BD est développée au sein d'une entreprise, pour son propre fonctionnement. Inversement, une **banque de données** est un ensemble de données, propres à un domaine d'application, que des "producteurs" réunissent pour ensuite en commercialiser l'usage vers le public extérieur.

Exemple :

Les banques de données juridiques, économiques, médicales, des brevets, des propriétés des matériaux, etc.

La constitution et l'exploitation des banques de données font appel à des techniques spécifiques, différentes de celles des bases de données, seules étudiées dans ce cours.

1.1.3 Fichier

Tout système d'exploitation d'un ordinateur contient un Système de Gestion de Fichiers (SGF) spécifique.

1.2 Cycle de vie d'une BD

On appelle **cycle de vie d'une BD** la suite des phases conception, implantation, utilisation.

- **La conception d'une BD** représente la phase d'analyse qui aboutit à déterminer le futur contenu de la base.

- **L'implantation de la BD** consiste à décrire la BD dans le langage du SGBD et de construire une première version de la BD. La description de la BD dans le langage du SGBD se fait au moyen d'un langage symbolique, spécifique du logiciel choisi, que l'on appelle **Langage de Description de Données** (LDD).

- **L'utilisation de la BD** se fait au moyen d'un langage, dit **Langage de Manipulation de Données** (LMD), qui permet d'exprimer aussi bien les requêtes d'interrogation (pour obtenir des informations contenues dans la base) que des requêtes de mise à jour (pour ajouter de nouvelles informations, supprimer des informations périmées, modifier le contenu des informations).

1.3 Modèles de données et schémas de BD

Au cours des différentes phases de la vie d'une BD, plusieurs descriptions sont successivement élaborées, chacune répondant à un objectif déterminé et complémentaire. Dans l'état actuel, ces descriptions ne peuvent être faites avec le langage naturel (en français, par exemple): celui-ci est trop ambigu et encore trop difficile à comprendre par un ordinateur. On fait donc appel à un langage formel, basé sur un certain nombre de concepts bien établis. Par exemple, les concepts d'**objet**, de **lien**, de **propriété**.

1.3.1 Modèle de données d'une BD

On appelle modèle de données l'ensemble des concepts qui permettent la description de données d'une base et les règles d'utilisation de ces concepts [1].

Un modèle de base de données a pour objectif de décrire la structure logique d'une base de données, il permet de structurer les informations et activités d'une organisation : données, traitements, et flux d'informations entre entités.

La modélisation des données est une représentation abstraite, elle a pour objet d'identifier les entités logiques et les dépendances logiques entre ces entités. Le modèle de données ne doit pas seulement définir la structure de données, mais aussi la sémantique des entités identifiées. Le résultat de la modélisation des données d'une BD est un schéma de la BD. Donc le processus de création d'un schéma de la BD s'appelle la modélisation des données.

Il existe de nombreux modèles de bases de données. Parmi ces modèles nous citons :

- Le modèle hiérarchique (utilisé par les systèmes de gestion d'information d'IBM dans les années 60 et 70, il est aujourd'hui majoritairement disparu).
- Le modèle réseau (extension du modèle hiérarchique, il a été utilisé dans les années 70).
- Le modèle relationnel (introduit en 1970, il se base sur l'algèbre relationnelle).

- Le modèle orientée objet (il se base sur le paradigme de l'orienté objet)

- Le modèle relationnel-objet (est un modèle hybride, il associe la simplicité du modèle relationnel à certaines des fonctionnalités avancées du modèle orientée objet.).

- Les modèles NoSQL (comme : le modèle orienté graphe, le modèle multi-valeur, le modèle orienté document).

Dans le cadre de ce cours, nous nous intéressons au modèle relationnel.

1.3.2 Schéma d'une BD

On appelle schéma d'une BD l'expression de la description de la BD en employant un modèle de données [2].

1.4 Architecture d'un SGBD

Un SGBD se compose de deux interfaces (Voir figure 1.1).

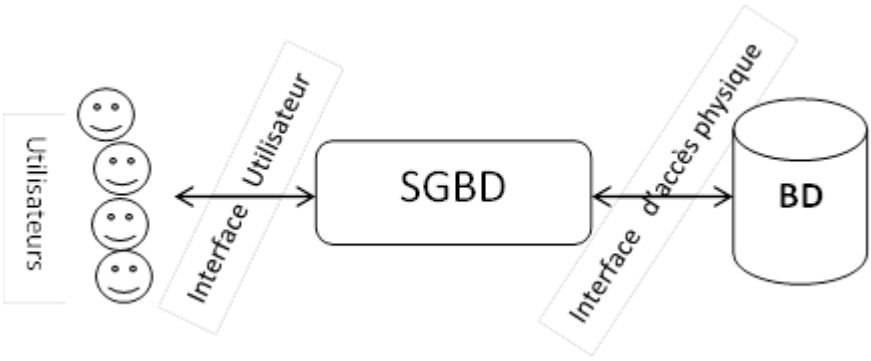


Figure 1.1 : Les interfaces d'un SGBD

- **L'interface utilisateur** permet aux utilisateurs d'exprimer des requêtes: soit pour définir le contenu de la BD (avec le LDD), soit pour interroger la BD (en extraire des informations), soit enfin pour apporter des modifications à ce qui a été enregistré (avec le LMD).

- **L'interface d'accès physique** permet au SGBD d'accéder aux données sur les supports (disques, etc.).

Un SGBD étant utilisé simultanément par plusieurs utilisateurs, il doit résoudre plusieurs problèmes internes de coordination de ses actions, de cohérence et de contrôle du bon déroulement de ses activités.

1.5 Les fonctions d'un SGBD

D'un point de vue fonctionnel, les fonctions d'un SGBD sont les suivantes :

- Supporter les concepts définis au niveau du modèle de données.
- Assurer et rendre transparent le partage des données entre différents utilisateurs.
- Assurer la confidentialité des données.
- Assurer le respect des règles de cohérence définies sur les données.
- Fournir différents langages d'accès selon le profil de l'utilisateur.
- Etre résistant aux pannes.
- Posséder une capacité de stockage élevée : Exa-octet (10^{18}) voire de zettaoctet (10^{21}).
- Pouvoir répondre à des requêtes avec un niveau de performances adapté.
- Fournir des facilités pour la gestion des métadonnées.

1.6 Conception d'une BD

La démarche proposée pour la conception d'une BD comprend deux phases (voir figure 1.2) :

- l'analyse conceptuelle,
- la production de la BDD.

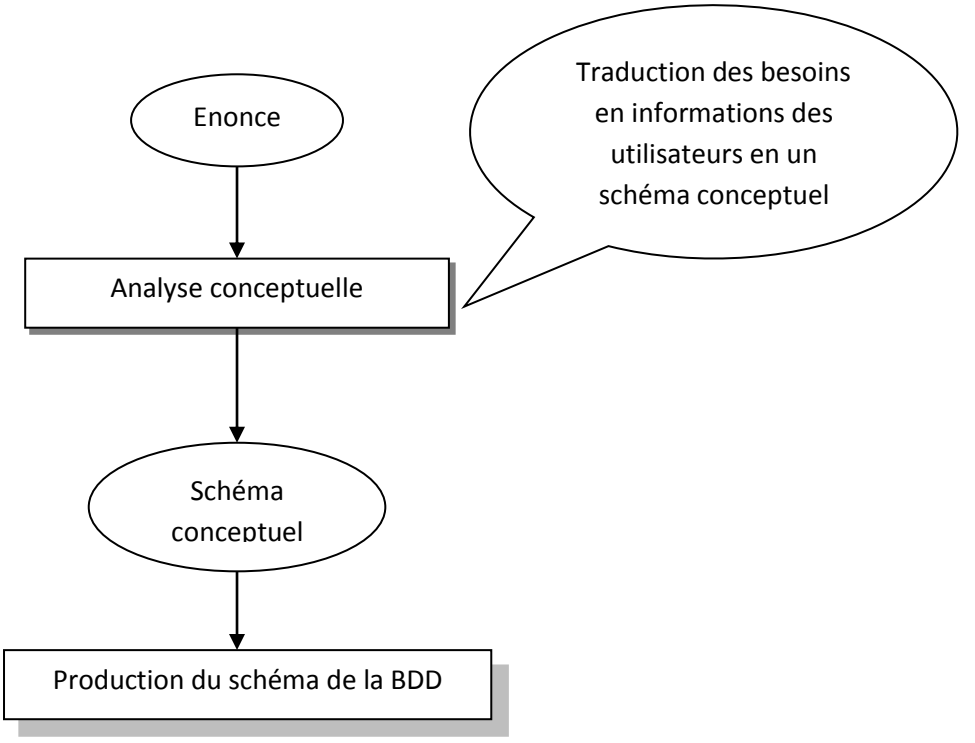
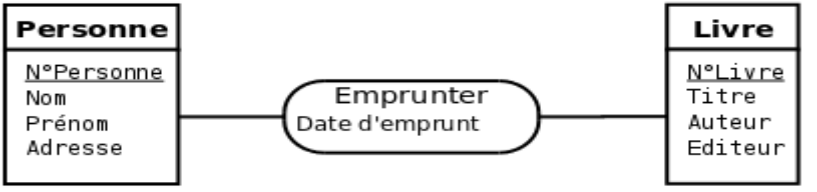


Figure 1.2 Structure générale de la méthode de construction d'une BDD

Exemples de modèle conceptuel EA

Exemple 1 : Personne / Emprunter / Livre



Exemple 2 : Etudiants / Cours / Profs

1.7 Conclusion

Dans ce chapitre, nous avons défini les concepts fondamentaux liés au domaine des BD. Il est très important alors de faire la différence entre une BD et une banque de données avant d'étudier le cycle de vie d'une BD qui est conçu comme une suite des phases

conception, implantation et utilisation d'une BD. De la phase de conception dépond tout le reste du cycle de vie de la BD, à ce stade, concevoir un modèle de BD qui répond aux besoins spécifiques des utilisateurs est donc une tâche non triviale. Vu sa grande simplicité, le modèle relationnel est certainement le plus utilisé actuellement.

Dans la communauté de modélisation des bases de données relationnelles[2], le modèle entité-association est utilisé pour établir un modèle de données spécifique au domaine. Le plus important est de savoir si le SGBD que nous utilisons prend en charge le modèle que nous avons construit.

Le modèle relationnel, organise les données dans des tables, que l'on appelle aussi des relations, chacune de ces relations se compose de colonnes et de lignes. Chaque colonne contient un attribut, comme le nom, le code postal, la date de naissance ou le salaire.

Une base de données relationnelle, permet de stocker et de retrouver les données propres à un système d'information en rapport avec une activité dans une entreprise.

Nous pouvons parler de trois générations de SGBD.

Avant 1962 : application, fichier,
Après 1962 : naissance de bases de données,

---

[2] Dans la communauté de la programmation orientée objet, on préfère le langage UML (Unified Modeling Language) pour la création de modèles de données.

Avant 1970 : 1re génération de SGBD (SGBD réseau, hiérarchique,
Après 1970 : 2ème génération : bases de données relationnelles,

Dans le cadre de ce cours, nous nous intéressons au modèle relationnel qui fera l'objet du chapitre suivant.

Chapitre 2

Le modèle relationnel

2.1 Introduction

Le modèle relationnel a été proposé par **Edgar Frank Codd**[3] (23 août 1923 - 18 avril 2003) en 1970. Il est considéré comme le plus simple des modèles (due à la vision tabulaire des données très intuitive), et il résulte de bases formelles issues de la théorie mathématique des ensembles.

Les objectifs du modèle relationnel étaient différents de ceux des modèles réseau et hiérarchique. Parmi les lacunes de ces modèles auxquelles **Edgar Frank Codd** souhaitait apporter une solution nous en retenons deux :

- Permettre un haut degré d'indépendance entre les applications (programmes, interfaces) et la représentation interne des données (fichiers, chemins d'accès).

- Etablir une base solide pour traiter les problèmes de cohérence et de redondance des données.

[3] https://fr.wikipedia.org/wiki/Edgar_Frank_Codd

Le modèle relationnel présente également de nombreux avantages dus au fait qu'il soit basé sur la théorie des ensembles : Langage de manipulation des données ensemblistes grâce à l'algèbre relationnelle et grâce à des langages assertionnels qui permettent de spécifier ce que l'on souhaite obtenir sans dire comment l'obtenir. Le SGBD est responsable de la politique d'exécution des requêtes.

2.2 Concepts du modèle relationnel

2.2.1 Domaine

Définition

Un domaine D est un ensemble de valeurs caractérisé par un nom. Du point de vue du modèle relationnel, chaque valeur du domaine est atomique et donc indivisible [1] [2].

Cette notion permet de définir les ensembles de départ. Un domaine peut être défini en extension en donnant la liste des valeurs composantes ou en compréhension en définissant une propriété caractéristique du domaine.

Exemple

COULEUR = {jaune; vert; rouge; bleu;}
ABONNE = {Personne possédant une carte d'abonné valide pour l'année en cours}

2.2.2 Schéma de relation

Définition

Un schéma de relation R, dénoté R(A1:D1, A2:D2, ..., An:Dn) est un ensemble d'attributs. Chaque attribut Ai est le nom d'un rôle joué par son domaine Di dans le schéma de relation R [1][2].

Un schéma de relation R est utilisé pour décrire une relation.

Exemple

Le schéma de la relation ABONNE est:

```
ABONNE (Num_Abonné : entier, Nom : chaîne(30), Prénom :
chaîne(40), Rue ...)
```

Définir un schéma de relation revient à spécifier un nouveau type de données équivalent à un type RECORD en langage Pascal. Le modèle relationnel n'autorise qu'un seul niveau de structure. Il n'est pas possible par exemple de définir un attribut Adresse qui se décompose en Rue, Ville, Code_Postal.

Remarque : On note généralement

R(A1, A2, …, An) au lieu de *R(A1:D1, A2:D2, ..., An:Dn)*

2.2.3 Relation

Définition

Une relation r dénotée r(R) du schéma de relation R(A1:D1, A2:D2, ..., An:Dn) est un ensemble d'enregistrements. Chaque enregistrement ei est une liste ordonnée de n valeurs ei = <v1, v2, ...,vn> où chaque vi est une valeur du domaine de l'attribut Ai ou une valeur nulle spéciale représentant l'absence d'information [1][2].

Notons que la présence de valeurs nulles dans une relation est souvent difficile à interpréter et rend plus complexe les manipulations.

Exemple

Relation ABONNE :

Abonné	Num_Abonné	Nom	Prénom	Rue	Ville	CodeP	Téléphone
	1	Mohamed	Ali	6, fg Agadir	Tlemcen	13000	04328889 9
	….						
	128	Fouad	Réda	20, rue AEK	Tlemcen	13000	04326998 8

Tuple ou occurrence de la relation

2.2.4 Clé d'une relation

Définition

Une clé **de relation** est un sous-ensemble d'attributs qui permet de caractériser tout enregistrement d'une relation *[1]*[2].

Par définition, une relation est un ensemble d'enregistrements et il ne peut donc pas y avoir deux enregistrements strictement identiques dans la même relation. Il existe généralement un sous-ensemble **SC** d'attributs d'un schéma de relation R pour lequel deux enregistrements de toute relation r(R) ne peuvent avoir la même combinaison de valeurs pour ces attributs.

$$\forall\ t_1, t_2 \in r(R),\ t_1[SC] \# t_2[SC].$$

Tout ensemble d'attributs vérifiant cette propriété est appelé **superclé** du schéma R. Il existe au moins une superclé qui est l'ensemble de tous les attributs.

Une **clé C d'un schéma R** est une superclé ayant la propriété suivante :

Si on enlève un attribut à C alors C n'est plus une superclé.
De manière informelle, une **clé** est un ensemble minimum d'attributs dont la connaissance des valeurs permet d'identifier un enregistrement unique de la relation considérée. Une clé est invariante dans le temps.

En général, il existe plusieurs clés pour une même relation R. Parmi les clés possibles, on choisit une clé qui sera appelée clé primaire. Lors de la définition d'un schéma cette clé est mise en évidence (soulignement).

Exemple

Clés de ABONNE (Num Abonné, Nom, Prenom, Rue, Ville, CodeP, Telephone)

Superclés : [Num Abonné, Nom], ….

Clé : Num Abonné (peut être le numéro de la carte d'identité nationale)

Clé primaire : Num Abonné {choisi parmi les clés}

2.2.5 Schéma de base de données et contraintes d'intégrité

Un **schéma de base de données relationnel** S est un ensemble de schémas de relation S = {R_1, R_2, …, R_p} et un ensemble de contraintes d'intégrité (CI) *[1]* [2].

Une **contrainte d'intégrité** est une propriété du schéma, invariante dans le temps *[1]* [2].

Il existe différents types de contraintes d'intégrité:

- liées au modèle (pas de doublons dans une relation.);
- de domaine (nb_heure < 100; pas de valeur nulle pour la clé primaire);
- référentielles dites de clé étrangère qui impose que la valeur d'attribut de la relation r1 apparaissent comme valeur de clé dans une autre relation r2.

Exemple

ABONNE (Num_Abonné, Nom, Prenom, Rue, Ville, CodeP, Telephone)

Prêter (Num_Abonné, Code_Ouvrage, Date_Prêt) : Num_Abonné doit être présent dans Prêter.

2.2.6 Instance de base de données relationnelle

Une instance de base de données relationnelle BD est un ensemble d'instances de relations *[1]* [2].

BD = $\{r_1, r_2, ..., r_n\}$ ou chaque r_i respecte les contraintes d'intégrité.

2.3 Algèbre relationnelle

L'algèbre relationnelle est une collection d'opérations permettant d'opérer sur les concepts du modèle relationnel. Elle permet par exemple de

sélectionner certains enregistrements d'une relation satisfaisant une condition ou encore de regrouper des enregistrements de relations différentes. Le résultat de toute opération de l'algèbre est une nouvelle relation. Cette propriété implique notamment qu'il n'y a pas de doublons dans le résultat et permet l'écriture d'expressions de calcul. Etant donnée, que le modèle relationnel est basé sur la théorie des ensembles, l'algèbre relationnelle utilise les opérateurs classiques de manipulation des ensembles (union, intersection, différence et produit cartésien) et introduit des opérateurs propres aux bases de données (sélection, projection, jointure, division).

2.3.1 Opérateurs ensemblistes

2.3.1.1 Union compatible

Deux relations sont union-compatibles si elles ont le même nombre d'attribut et que ceux-ci ont le même domaine.

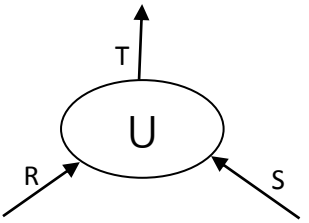
$\cup$ et $\cap$ nécessitent que les relations soient union-compatibles.

2.3.1.2 Opérateur union (U)

Définition

Soient R et S, deux relations de schémas respectifs X et Y. Les schémas X et Y doivent être union-compatibles c'est à dire posséder le même nombre d'attributs et que ceux-ci soient de même domaine. *L'union des deux relations, R $\cup$ S produit une nouvelle relation de schéma identique à R possédant les enregistrements appartenant à R ou à S ou aux deux relations [1].*

Notation : T = R $\cup$ S ; Ou T = UNION(R,S)



Représentation Graphique :

Exemple

R	A	B
	X_1	Y_1
	X_2	Y_2
	X_3	Y_3

S	C	D
	X_4	Y_4
	X_1	Y_1
	X_2	Y_3

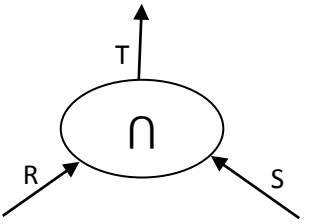
R ∪ S	A	B
	X_1	Y_1
	X_2	Y_2
	X_3	Y_3
	X_4	Y_4
	X_2	Y_3

2.3.1.3 Opérateur intersection (⋂)

Définition

Soient R et S, deux relations de schémas respectifs X et Y. Les schémas X et Y doivent être union-compatibles. L'intersection des deux relations R ⋂ S produit une nouvelle relation de schéma identique à R possédant les enregistrements appartenant conjointement à R et à S *[1]*.

Notation : T= R ⋂ S ou T = INTERSECT (R,S)



Représentation graphique :

Exemple

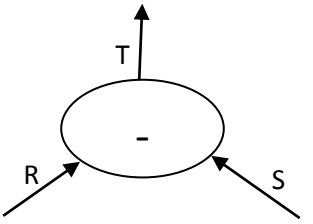
R ∩ S	A	B
	X_1	Y_1

2.3.1.4 Opérateur différence (-)

Définition

Soient R et S, deux relations de schémas respectifs X et Y. Les schémas X et Y doivent être union-compatibles. La différence des deux relations R - S produit une nouvelle relation de schéma identique à R possédant les enregistrements présents dans R mais pas dans S *[1]*.

Notation : T= R-S ; OU T = MINUS(R,S)



Représentation Graphique :

Exemple

R - S	A	B
	X_2	Y_2
	X_3	Y_3

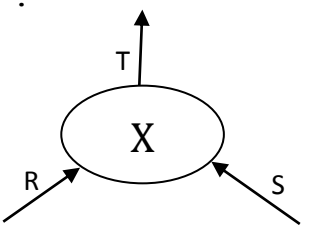
2.3.1.5 Opérateur produit cartésien (X)

Définition

Soient R et S, deux relations de schémas respectifs X et Y. Les schémas X et Y doivent être disjoints c'est à dire ne pas avoir d'attributs communs. *Le produit cartésien des deux relations R X S produit une nouvelle relation de schéma T égal à l'union des schémas R et S et possédant comme enregistrements, la concaténation des enregistrements de R avec ceux de S [1].*

Notation : T= R X S ; OU T = PRODUCT(R,S) OU T = TIMES(R,S)

Représentation Graphique :



Exemple

R X S	A	B	C	D
	X_1	Y_1	X_4	Y_4

	X_1	Y_1	X_1	Y_1
	X_1	Y_1	X_2	Y_3
	X_2	Y_2	X_4	Y_4
	X_2	Y_2	X_1	Y_1
	X_2	Y_2	X_2	Y_3
	X_3	Y_3	X_4	Y_4
	X_3	Y_3	X_1	Y_1
	X_3	Y_3	X_2	Y_3

2.3.2 Opérateurs Bases de Données

2.3.2.1 Sélection (σ)

Définition

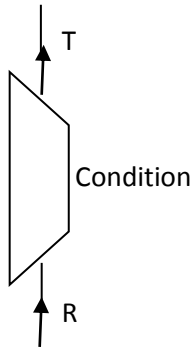
La sélection prend en entrée une relation R de schéma X et produit en sortie une nouvelle relation de schéma X ayant comme enregistrements ceux de R satisfaisant la condition de sélection. La condition de sélection utilise les opérateurs de comparaison (<; <=; >; =>; =; !=), les connecteurs logiques (et ou non) et les parenthèses *[1]*.

Notation : T= σ Condition(R);OU T = R[Condition]

OU T = RESTRICT(R/Condition)

OU T = SELECT(R/Condition)

Représentation Graphique :



Exemple

Soit la relation LIVRE

LIVRE	CodeOuv	Titre	Genre	Editeur	Collection
	5894	Les BDD	Informatique	E1	Classique
	4774	Maths	Maths	E2	Sciences
	2560	SQL	Informatique	E3	Best-seller
	8541	La main verte	Science-fiction	E4	Nuit

Q1= σ (Genre='Informatique' ou Editeur='E1') (LIVRE)

Q1	CodeOuv	Titre	Genre	Editeur	Collection
	5894	Les BDD	Informatique	E1	Classic
	2560	SQL	Informatique	E3	Best-seller

2.3.2.2 Projection (π)

Définition

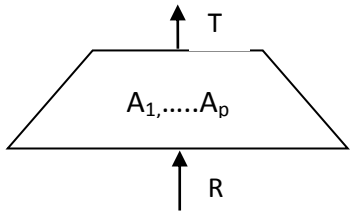
La projection prend en entrée une relation R de schéma X et produit en sortie une nouvelle relation de schéma A_1; A_2; ...; A_n (schéma inclu dans X) ayant comme enregistrements ceux de R restreints au sous-schéma A_1; A_2; ...; A_n *[1]*.

Notation : T= π A1,A2,..,Ap(R) ;

OU T= R{ A1,A2,..,Ap }

OU T = PROJECT(R/ A1,A2,..,Ap)

Représentation Graphique :



Exemple

Q2 = π Titre (LIVRE)

Q2	Titre
	Les BDD
	Maths
	SQL
	La main verte

2.3.2.3 Jointure ($\bowtie$)

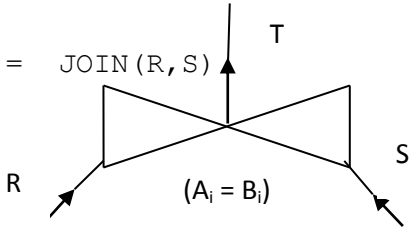
Définition

Soient R et S deux relations de schéma respectif X et Y. Il faut que les schémas X et Y possèdent une intersection Z non vide. La relation résultat a un schéma qui est l'union des deux schémas X et Y et comme enregistrements la concaténation des enregistrements de R avec ceux de S s'ils ont la même valeur pour les attributs communs *[1][2]*.

$R(A_1,.....A_p)$, $S(B_1,.....B_n)$

Notation : T= $R \bowtie S$; OU T = JOIN(R,S)

Représentation Graphique :



Exemple

R	X	Y	Z
	X_1	Y_1	Z_1
	X_2	Y_2	Z_2
	X_1	Y_1	Z_3
	X_3	Y_3	Z_4

S	Y	Z	U
	Y_1	Z_1	U_1
	Y_1	Z_1	U_2
	Y_1	Z_3	U_4
	Y_3	Z_8	U_8

R ⋈ S	X	Y	Z	U
	X_1	Y_1	Z_1	U_1
	X_1	Y_1	Z_1	U_2
	X_1	Y_1	Z_3	U_4

2.3.2.4 Division (÷)

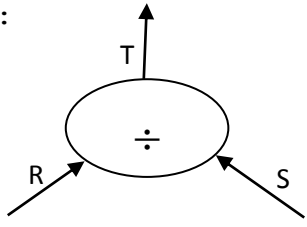
Définition

La relation résultat est définie sur le schéma X et comprend tous les n-uplets dont la concaténation avec tous les n-uplets de S appartiennent à R *[1]*.

R est défini sur un shéma X;
S est défini sur un schéma Y.

Notation : T= R ÷ S; OU T =DIVISION(R,S)

Représentation Graphique :



Exemple

Q4 : "Quels sont les étudiants qui pratiquent tous les sports ?"

Pratiquer	Etudiants	Sports
	Med	Foot
	Salim	Foot
	Med	Hand
	Sami	Foot
	Sami	Judo
	Sami	Hand
	Sami	Basket

Sports	Sports
	Foot
	Judo
	Hand
	Basket

Q4	Etudiants
	Sami

2.4 Normalisation d'une relation

2.4.1 Introduction

Les formes normales permettent d'éviter la redondance, source d'anomalies. Il existe 5 formes normales principales. Plus le niveau de normalisation est élevé, plus le modèle est exempte de redondances. Un type-entité ou un type-association en forme normale de niveau n est automatiquement en forme normale de niveau n-1. Une modélisation rigoureuse permet généralement d'aboutir directement à des type-entités et type-associations en forme normale de Boyce-Codd.

2.4.2 Relation universelle

Table unique dont le schéma est composé par union de tous les attributs des tables constituant la base.

2.4.3 Décomposition d'une relation

C'est le remplacement d'une relation R (A_1, A_2, … A_n) par une collection de relations R_1, R_2, … R_n obtenu par des projections de R sur des sous-ensembles d'attributs dont l'union contient tous les attributs de R.

2.4.4 Décomposition sans perte

C'est la décomposition d'une relation R en R_1, R_2, … R_n telle que pour toute extension de R, on ait :

$$R = R_1 \bowtie R_2 \bowtie \dots R_n$$

La conception d'une base de données relationnelle consiste à décomposer la relation universelle sans perte d'information.

2.4.5 Dépendances fonctionnelles (DF)

La notion de dépendance fonctionnelle permet de caractériser des relations pouvant être décomposées sans perte d'informations.

2.4.5.1 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 →Y (X détermine Y, ou Y dépond fonctionnellement de X) si pour toute extension r de R, pour tout tuple t_1 et t_2 de r, on a :

$$\pi_X(t_1) = \pi_X(t_2) \Rightarrow \pi_Y(t_1) = \pi_Y(t_2)$$

Exemple

Soit la relation

Véhicule (NV, Type, Marque, Puissance, Couleur)

On a : NV →Couleur

Type → Marque

Type → Puissance

(Type, Marque) → Puissance

Puissance ↛ Type et Type ↛ Couleur

2.4.5.2 Propriété des DFs

Réflexivité : $Y \subseteq X \Rightarrow X \rightarrow Y$

Augmentation : $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$

Transitivité : $X \rightarrow Y$ et $Y \rightarrow Z \Rightarrow X \rightarrow Z$

Union : $X \rightarrow Y$ et $X \rightarrow Z \Rightarrow X \rightarrow YZ$

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

Décomposition : $X \rightarrow Y$ et $Z \subseteq Y \Rightarrow X \rightarrow Z$

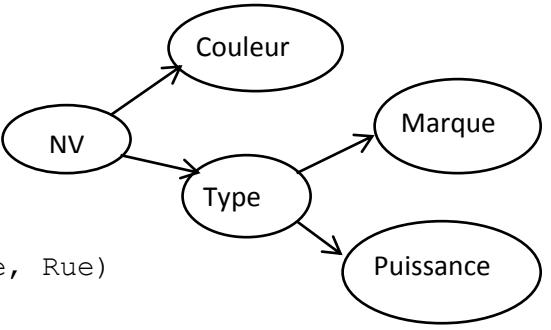
2.4.5.3 Dépendance Fonctionnelle Elémentaire (DFE)

Dépendance fonctionnelle de la forme X→ A où A est un attribut unique n'appartenant pas à X et où il n'existe pas X'⊂ X tel que X' → A.

2.4.5.4 Graphe de DF

Soit un ensemble F de DFE. Si tous les attributs gauches sont uniques, il est possible de visualiser cet ensemble de DF par un graphe appelé graphe de DF.

Exemple de graphe de DF

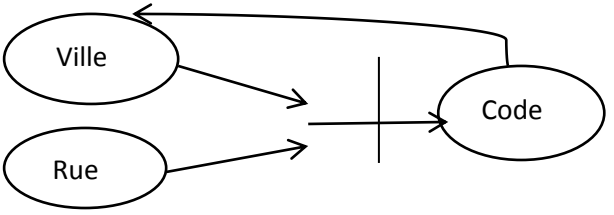


Un autre exemple

Code postal (Code, Ville, Rue)

(Ville, Rue) ⟶ Code

Code ⟶ Ville



2.4.5.5 Fermeture transitive (F+) et couverture minimale

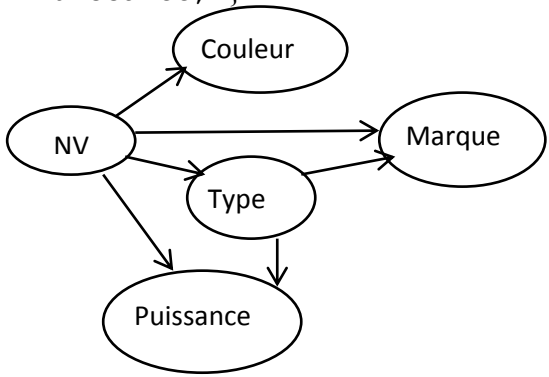
La fermeture transitive d'un ensemble F de DFE est l'ensemble des DFE considérées enrichi de toutes les DFE déduites par transitivité.

Exemple

F = { NV→ Type, Type → Marque, Type → Puissance, NV → Couleur}

F^+ **= F U** { NV → Marque, NV → Puissance, }

Le graphe de F^+ est :



Deux ensembles de DF sont équivalents s'ils ont la même fermeture transitive.

La couverture minimale est l'ensemble F de DFE vérifiant les propriétés suivant :

a) Aucune DF dans F n'est redondante

Autrement, pour toute DF f de F, $F - f$ n'est pas équivalent à F.

b) Toute DFE des attributs est dans F^+.

Il a été montré que tout ensemble de DF a une couverture minimale qui n'est en général pas unique.

Exemple

F = { NV→ Type, Type → Marque, Type → Puissance, NV → Couleur}

Est une couverture minimale pour l'ensemble des DFs de véhicule.

Remarque

La couverture minimale constitue un élément essentiel pour composer des relations sans pertes d'informations.

2.4.6 Clé d'une relation

Est le sous-ensemble X des attributs d'une relation

R (A_1, A_2, … A_n) tel que

a) X → A_1, A_2, … A_n

b) Il n'existe pas de sous-ensemble Y ∈ X tel que :

Y → A1, A2, … An

✓ Une clé est un ensemble minimal d'attributs qui détermine tous les autres.

✓ Un ensemble d'attributs qui inclut une clé est appelé superclé.

Par exemple, NV est une clé et (NV, Type) est une superclé (est non une clé).

Remarques

- Il peut y avoir plusieurs clés pour une même relation : on choisit en général une comme clé primaire.
- Une clé candidate est une clé quelconque.

2.4.7 Les formes normales (FN)

2.4.7.1 Première FN (1FN)

Une relation est en 1FN si tout attribut contient une valeur atomique.

2.4.7.2 Deuxième FN (2FN)

Une relation est en 2FN si et seulement si :

a) Elle est en 1FN.

b) Tout attribut n'appartenant pas à une clé ne dépend pas d'une partie d'une clé.

2.4.7.3 Troisième FN (3FN)

Une relation est en 3FN si et seulement si :

a) Elle est en 2FN.

b) Tout attribut n'appartenant pas à une clé ne dépend pas d'un autre attribut non clé.

Exemple

Véhicule (NV, Type, Marque, Puissance, Couleur)

Cette relation n'est pas en 3FN car :
L'attribut non clé Type → Marque et Type → Puissance.
Cette relation peut être décomposée en deux relations :
Véhicule (NV, Type, Couleur) et Modèle (Type, Marque, Puissance)

Remarque

Toute relation a au moins une décomposition en 3FN telle que :

a) La décomposition préserve les DF
b) La décomposition est sans perte.

2.4.7.4 Forme normale de Boyce-Codd (BCNF)

Une relation est *en forme normale de Boyce-Codd si et seulement si :*

a) Elle est en 3FN.
b) Aucun attribut faisant partie de la clé dépend d'un attribut ne faisant pas partie de la clé.

Remarque

Un modèle en forme normale de Boyce-Codd est considéré comme étant de qualité suffisante pour une implantation.

2.4.7.5 Autres formes normales

Il existe d'autres formes normales. La quatrième et la cinquième forme normale. Ils ne font pas l'objet de ce cours, vous pouvez consulter la référence [1] pour la définition de ces formes.

2.5 Exercices avec solutions

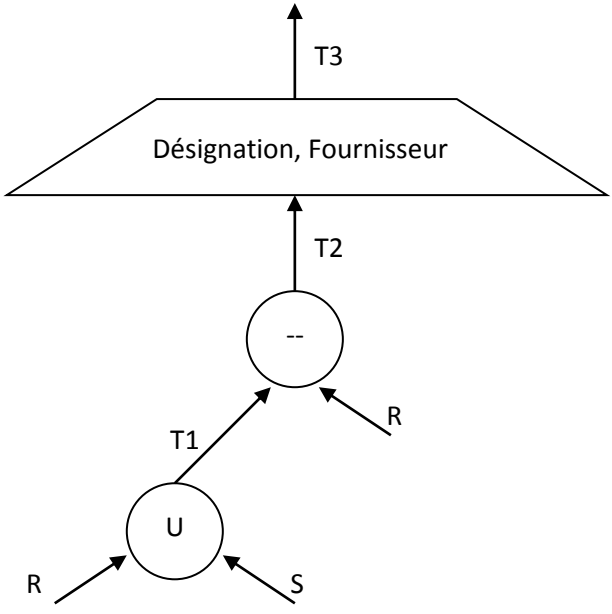
2.5.1 Exercice 1

Soient R et S les deux relations suivantes

R	NProduit	Désignation	Fournisseur	Date	Qté
	A20	Café	F1	13/01/2007	150
	A25	Thé	F1	13/01/2006	120
	A30	Riz	F1	15/01/2007	130

S	NProduit	Désignation	Fournisseur	Date	Qté
	A20	Café	F1	20/01/2007	150
	F35	Semoule	F2	21/04/2006	120
	A30	Riz	F3	15/01/2007	130

Effectuez les opérations de l'arbre relationnel suivant :



Solution exercice 1

✓ T1 = R U S

T1	NProduit	Désignation	Fournisseur	Date	Qté
	A20	Café	F1	13/01/2007	150
	A25	Thé	F1	13/01/2006	120
	A30	Riz	F1	15/01/2007	130
	A20	Café	F1	20/01/2007	150
	F35	Semoule	F2	21/04/2006	120
	A30	Riz	F3	15/01/2007	130

✓ T2 = T1 - R = S même relation que S

✓ T3 = π Désignation, Fournisseur (T2)

T3	Désignation	Fournisseur
	Café	F1
	Semoule	F2
	Riz	F3

2.5.2 Exercice 2

Soient R et S les deux relations suivantes

R	Num_SS	Nom	DN	Code Fonction	Num_Véhicule
	10	Mohamed	12/12/1980	21	2
	15	Omar	25/10/1981	25	5
	20	Fatima	13/11/1981	12	26
	10	Mohamed	12/12/1980	21	4

Avec Num_SS : Numéro de sécurité sociale

S	Num_Véhicule	Marque	Puissance	Couleur
	2	Renault	5	Bleu
	3	Peugeot	5	Noir

4	Citroën	6	Jaune
26	Fiat	5	Bleu
5	Handa	5	Noir

Effectuez les opérations suivantes :

- T1 = R ⋈ S
- T2 = π Num_SS, Nom, Marque (T1)
- T3 = π Nom (T1)

Solution exercice 2

✓ T1 = R ⋈ S

T1	Num_SS	Nom	DN	Code Fonction	Num_Véhicule	Marque	Puissance	Couleur
	10	Mohamed	12/12/1980	21	2	Renault	5	Bleu
	15	Omar	25/10/1981	25	5	Honda	5	Noir
	20	Fatima	13/11/1981	12	26	Fiat	5	Bleu
	10	Mohamed	12/12/1980	21	4	Citroën	6	Jaune

✓ T2 = π Num_SS, Nom, Marque (T1)

T2	Num_SS	Nom	Marque
	10	Mohamed	Renault

15	Omar	Citroën
20	Fatima	Fiat
10	Mohamed	Handa

✓ T3 = π Nom (T1)

T3	Nom
	Mohamed
	Omar
	Fatima

2.5.3 Exercice 3

On souhaite créer une base de données concernant une entreprise. Une première étude a mis en évidence trois relations. Pour chacune des relations, la clé est soulignée.

EMPLOYE (NumEmp, Nom, Prénom, Adresse, Téléphone, Qualification)

SERVICE (NomService, Responsable, Téléphone)

PROJET (NomProjet, DateDeb, DateFin, NumEmp)

En considérant les possibilités offertes par ce schéma, répondre aux questions suivantes **en justifiant vos réponses par rapport au modèle relationnel** et par rapport à la sémantique intuitive des relations :

Question 1 :

Un employé peut-il avoir plusieurs qualifications ?

Question 2 :

Un employé peut-il faire plusieurs projets en même temps ?

Question 3 :

Une personne peut-elle être responsable de plusieurs services ?

Question 4 :

Un service peut-il avoir plusieurs responsables ?

Solution exercice 3

- ✓ Q1 : Non
- ✓ Q2 : Non
- ✓ Q3 : Oui
- ✓ Q4 : Oui

2.5.4 Exercice 4

Soit la base de données relationnelle de schéma :

U (NU, NomU, Ville)

P (NP, NomP, Couleur, Poids)

F (NF, NomF, Statut, Ville)

PUF (NP, NU, NF, Quantité)

NP référence P.NP

NU référence U.NU

NF référence F.NF

décrivant le fait que :

U: une usine est décrite par son numéro NU, son nom NomU, la ville Ville dans laquelle elle est située;

P: un produit est décrit par son numéro NP, son nom NomP, sa Couleur, son Poids;

F: un fournisseur est décrit par son numéro NF, son nom NomF, son Statut (fournisseur sous-traitant, fournisseur-client, ...), la Ville où il est domicilié;

PUF: le produit de numéro NP a été livré à l'usine de numéro NU par le fournisseur de numéro NF dans une Quantité donnée.

Exprimez en algèbre relationnelle les requêtes suivantes:

1) Donner le numéro, le nom et la ville de toutes les usines.
2) Donner le numéro, le nom et la ville de toutes les usines de Tlemcen.
3) Donner les numéros des fournisseurs qui approvisionnent l'usine no 1 en produit no 1.
4) Donner le nom et la couleur des produits livrés par le fournisseur no 1.
5) Donner les numéros des fournisseurs qui approvisionnent l'usine no 1 en un produit rouge.
6) Donner les noms des fournisseurs qui approvisionnent une usine de Tlemcen ou de Sidi Belabbes en un produit rouge.

7) Donner les numéros des produits livrés à une usine par un fournisseur de la même ville.

8) Donner les numéros des produits livrés à une usine de Tlemcen par un fournisseur de Tlemcen.

9) Donner les numéros des usines qui ont au moins un fournisseur qui n'est pas de la même ville.

10) Donner les numéros des fournisseurs qui approvisionnent à la fois les usines no 1 et no 2.

11) Donner les numéros des usines qui utilisent au moins un produit disponible chez le fournisseur no 3 (c'est-à-dire un produit qu'il livre mais pas nécessairement à cette usine).

12) Donner le numéro du produit le plus léger (les numéros si plusieurs produits ont ce même poids). En algèbre relationnelle où il n'existe pas de quantificateur universel, cette requête peut s'écrire en cherchant le complément du résultat : l'ensemble des produits qui ont un produit plus léger qu'eux.

13) Donner les numéros des usines qui ne reçoivent aucun produit rouge d'un fournisseur Tlemcenien.

14) Donner tous les triplets (VilleF, NP, VilleU) tels qu'un fournisseur de la première ville approvisionne une usine de la deuxième ville.

15) Même question qu'en 15, mais sans les triplets où les deux villes sont identiques.

16) Donner les numéros des produits qui sont livrés à toutes les usines de Tlemcen.

17) Donner les numéros des usines qui s'approvisionnent uniquement chez le fournisseur no 3.

Solution exercice 4

1. R1= π NU, NomU, Ville (U)

2. R2= π NU, NomU, Ville (σ Ville =″Tlemcen″ (U))

3. π NF(σ NU = 1 et NP = 1 (PUF)

4. t1 = σ NF = 1 (PUF), t2 = t1 $\bowtie_{t1.NP = P.NP}$ P

R4 = π NomP, Couleur (t2)

5. t1 = σ NU = 1 (PUF)

t2 = t1 $\bowtie_{t1.NP = P.NP}$ P

t3 = σ Couleur = ″rouge″ (t2)

R5 = π NF (t3)

6. t1 = PUF $\bowtie_{PNF.NP = P.NP}$ P

t2 = π Couleur = ″rouge″ (t1)

t3 = t2 $\bowtie_{t2.NU = U.NU}$ U

t4 = σ Ville =″Tlemcen″ ou Ville =″SBA″ (t3)

t5 = t4 $\bowtie_{t4.NF = F.NF}$ F

R6 = π Nom F (t5)

7. t1 = U ⋈ F
U. Ville = F. Ville

t2 = t1 ⋈ PUF
t.NF = PUF.NF

R7 = π NP (t2)

8. σ U. Ville = ″Tlemcen″ (PUF ⋈ U ⋈ F)

R = π NP (R1)

9. R1 = π U. Ville < > F. Ville (PUF ⋈ U ⋈ F)

R = π NU (R1)

10. R1 = π NF (σ NU = 1 (PUF))

R2 = π NF (σ NU = 2 (PUF))

R3 = R1 ∩ R2

11. R = π NU (σ NF = 3 (PUF))

12. R1 = P ⋈ P′
P.Poids < P′.Poids

R2 = π NP.Poids (R1)

R3 = π NP.Poids′ (R1)

R = π NP.Poids (R2-R3)

13. R1 = σ P. Couleur = ″rouge″ ∧ F. Ville = ″Tlemcen″ (PUF ⋈ P ⋈ F)

R2 = π NU (R1)

R3 = π NU (σ F. Ville = ″Tlemcen″ (PUF ⋈ F)

R13 = R3 – R2

Ou

R13 = π NU (σ P. Couleur = ″rouge″ ∧ F. Ville = ″Tlemcen″ (PUF ⋈ P ⋈ F)

14. R1 = PUF ⋈ U ⋈ F

R14 = π F. Ville , NP, U. Ville (R1)

15. R1 = σ F. Ville < > U. Ville (PUF ⋈ U ⋈ F)

R15 = π F. Ville , NP, U. Ville (R1)

17. T = σ NF = 3 (PUF)

T1 = π NU (T)

T2 = σ NF ≠ 3 (PUF)

T3 = π NU (T2)

R17 = T1 – T3

2.5.5 Exercice 5

Soit la base de données relationnelle, Employé, de schéma :

Employé(Nom,Prénom,DateNaissance,Adresse,N°Emp,Salaire,N°Dep,Supérieur)

Supérieur fait référence à : Employé.N°Emp

N°Dep fait référence à : Département.N°Dep

Département (NomD, N°Dep, Directeur)

Directeur fait référence à Employé.N°Emp

Projet (NomP, N°Pro, Lieu, N°Dep)

N°Dep fait référence à Département.N°Dep

Travaille (N°Emp, N°Pro, Heures)

N°Pro fait référence à Projet.N°Pro

N°EMP fait référence à Employé.N°Emp

Précision:

L'attribut "Supérieur" dans la relation "Employé" contient le numéro Emp du supérieur direct de l'employé. Chaque employé appartient à un département et travaille sur un ou plusieurs projets. Chaque projet est rattaché à un département qui peut être différent de celui des employés travaillant sur ce projet.

Exprimez en algèbre relationnelle les requêtes suivantes :

1) Date de naissance et adresse de Abdellah Mohammed.
2) Nom et adresse des employés qui travaillent au département de recherche.
3) Nom et prénom des employés dont le supérieur est Abdellah Mohammed.
4) Nom des employés qui travaillent plus de 10 heures sur un projet à Tlemcen.
5) Nom des projets sur lesquels travaillent Abdellah Mohammed et Abdelmadjid Salim . Attention le "et" du français signifie ici que

l'un ou l'autre, ou les deux, doivent travailler au projet.

6) Nom des projets sur lesquels travaillent à la fois Abdellah Mohammed et Abdelmadjid Salim.

7) Nom et prénom des employés qui ne travaillent sur aucun projet.

8) Numéro des projets qui ont au moins un participant de chaque département.

9) Nom des employés qui ne travaillent pas sur un projet à Tlemcen.

10) Nom des employés qui ne travaillent que sur des projets à Tlemcen.

Solution exercice 5

1) R1 = π DateNaissance, Adresse (Employé)

2) R2 = π Nom, Adresse (Employé $\bowtie$ Département_R)
(E. N°Emp = D. N°Emp)

Département_R = σ Nom D = ″ Recherche″ (Département)

3) t1 = σ Nom = ″Abdellah″ et Prénom = ″ Mohammed″ (Employé)

t2 = Employé $\bowtie$ t1
E. Supérieur = t1. N°Emp

R3 = π Nom, Prénom (t2)

4) t1 = σ lieu = ″Tlemcen″ (Projet)

t2 = σ Heure > 10 (Travail)

t3 = t1 ⋈ t2

t4 = (Employé ⋈ t3)

E. N°Emp = t3. N°Emp

R4 = π Nom (t4)

5) t1 = σ(Nom = ″Abdellah″ et Prénom = ″ Mohammed″)ou (Nom = ″ Abdelmadjid ″ et Prénom = ″ Salim ″) (Employé)

t2 = t1 ⋈ Travail

t1. N°Emp = T. N°Emp

t3 = t2 ⋈ Projet

t2. N°Pro = P. N°Pro

R5 = π Nom P (t3)

6) t1 = σ Nom = ″Abdellah″ et Prénom = ″ Mohammed″ (Employé)

t2 = t1 ⋈ Travail

t.. N°Emp = T. N°Emp

t3 = t2 ⋈ Projet

t2. N°Pro = P. N°Pro

t4 = π Nom P (t3)

t1′ = σ (Nom = ″ Abdelmadjid ″ et Prénom = ″ Salim ″) (Employé)

t2′ = t1′ ⋈ Travail

N°Emp

t3′ = t2′ ⋈ Projet

N°Pro

t4′ = π Nom P (t3′)

R6 = t4 ∩ t4′

7) t1 = π Nom Emp (Employé)

t2 = π Nom Emp (Travail)

t3 = t1 – t2

t4 = t3 ⋈ Employé
N°Emp = E. N°Emp

R7 = π Nom, Prénom (t4)

8) t1 = Travail ⋈ Employé.
N°Emp

t2 = π N°Pro, N°Dep (t1)

t3 = π N°Dep (Département)

R8 = t2 ÷ t3

9) t1 = σ lieu ≠ ″Tlemcen″ (Projet)

t2 = t1 ⋈ Travail
N°Pro

t3 = t2 ⋈ Employé.
N°Emp

R9= π Nom (t3)

10) t1 = Travail ⋈ Projet
N°Pro

t2 = t1 $\bowtie$ Employé.

N°Emp

t3 = σ lieu = ″Tlemcen″ (t2) ; t3′ = π N Emp, Nom (t3)

t4 = σ lieu ≠ ″Tlemcen″ (t2) ; t4′ = π N Emp, Nom (t4)

R10 = π Nom (t3′ - t4′)

2.6 Conclusion

Dans ce chapitre, nous avons introduit les concepts du modèle relationnel et décrit l'ensemble des opérateurs de l'algèbre relationnelle. Nous avons proposé vers la fin de ce chapitre des exercices corrigés.

Chapitre 3

Le langage SQL

3.1 Introduction

Le langage **SQL** (Structured Query Language) peut être considéré comme le langage d'accès normalisé aux bases de données. Il est aujourd'hui supporté par la plupart des produits commerciaux que ce soit par les systèmes de gestion de bases de données micro tel que « Access » ou par les produits plus professionnels tels que « Oracle ». Il a fait l'objet de plusieurs **normes ANSI/ISO** dont la plus répandue aujourd'hui est la norme SQL2 qui a été définie en 1992. Nous décrivons ici les principaux aspects de cette norme.

Le succès du langage SQL est dû essentiellement à sa simplicité et au fait qu'il s'appuie sur le schéma conceptuel pour énoncer des requêtes en laissant le SGBD responsable de la stratégie d'exécution. Le langage SQL propose un langage de requêtes ensembliste et assertionnel. Néanmoins, le langage SQL ne possède pas la puissance d'un langage de programmation : entrées/sorties, instructions conditionnelles, boucles et affectations. Pour certains traitements il est donc nécessaire de coupler le langage SQL avec un langage de programmation.

Le langage SQL comporte :

- **Une partie sur la définition des données** : le langage de définition des données **(LDD)** qui permet de définir des relations, des vues externes et des contraintes d'intégrité;

- **Une partie sur les requêtes** : le langage de manipulation des données **(LMD)** qui permet d'interroger une base de données sous forme déclarative sans se préoccuper de l'organisation physique des données;

- **Une partie sur le contrôle des données** : le langage de contrôle des données **(LCD)** qui permet de contrôler la sécurité et les accès aux données.

Nous présentons ici les principaux aspects du langage SQL. La syntaxe de ce langage est tirée de [1][2] et [5].

3.2 Le langage de définition des données (LDD)

Le **langage de définition des données** permet de définir et de manipuler les concepts du modèle relationnel liés au schéma : relation, attribut, clé, contraintes d'intégrité, vues ainsi que certains éléments liés à l'administration de la base de données : index, droits des utilisateurs.

Les types de données disponibles dépendent du SGBD mais on retrouve généralement les types entier, réel, chaînes de caractères et date. Les principaux types de données disponibles en SQL sont : SMALLINT, INTEGER, DECIMAL, FLOAT, DOUBLE, DATE, TIME, TIMESTAMP, CHAR et VARCHAR.

3.2.1 Les relations

- **Créer une base de données**

Syntaxe

```
CREATE DATABASE <nom_de_base>
```

Exemple (Créer une base de données « Hopital »)

```
CREATE DATABASE `Hopital` ;
```

- **Créer une relation (Table)**

Syntaxe

```
CREATE TABLE <nom_de_relation> (
<nom_d'attribut> <type_de_données> [NOT NULL] [, ...]
[, PRIMARY KEY (<nom_d'attribut> [, ...] ) ]
[, FOREIGN KEY (<nom_d'attribut>) REFERENCES
<nom_de_relation> [, ...]]
[, ON DELETE {RESTRICT | NO ACTION | SET NULL | CASCADE}
])
[CHECK (<condition>)]
```

Exemple (Créer la table Patient avec comme attribut : Num, Nom, Prénom et Adresse)

```
CREATE TABLE `Patient` (
`Num_Patient` INT NOT NULL ,
`Nom` VARCHAR( 25 ) NOT NULL ,
`Prenom` VARCHAR( 25 ) NOT NULL ,
`Adresse` VARCHAR( 45 ) NOT NULL ) ;
```

- **Modifier le schéma de la relation.**

Syntaxe

ALTER TABLE <nom_de_relation> **ADD(** <nom_d'attribut> <type_de_données> **)**

Exemple (Ajouter la colonne EMAIL à la table Patient)

ALTER TABLE `Patient` ADD `EMAIL` VARCHAR(30) NOT NULL ;

- **Supprimer une relation d'un schéma de base de données.**

Syntaxe

DROP TABLE <nom_de_relation> [CASCADE CONSTRAINTS]

Exemple (Supprimer la relation Patient)

DROP TABLE `Patient`;

La clause CASCADE CONSTRAINS Permet de supprimer toutes les contraintes d'intégrités référentielles qui réfèrent aux clés (PRIMARY KEY et unique) de la relation supprimée.

Cette clause concerne les relations où des contraintes de clés étrangères ont été spécifiées. Par exemple, la suppression de la relation 'Patient' remet en cause la contrainte référentielle sur Num_Patient spécifiée dans la relation Consultation.

Exemple (Supprimer la relation Patient avec la clause CASCADE CONSTRAINS)

DROP TABLE `Patient` CASCADE CONSTRAINTS;

3.2.2 Les index

La création d'un **index** permet de réduire les temps de recherche.

La création d'index ne fait pas partie de la norme ANSI SQL. Néanmoins, la plupart des produits SQL supportent la création d'index. L'effet de cette commande est la création d'un index en général de type B-arbre sur les attributs spécifiés avec les clés triées soit par ordre ascendant (ASC) ou descendant (DESC). Si le mot clé UNIQUE est utilisé, le SGBD interdit les doublons.

- **Crée un index.**

Syntaxe

```
CREATE [UNIQUE] INDEX <nom_de_l'index>

ON <nom_de_relation> (<nom_d'attribut> [{ASC | DESC}]
[, ...])
```

Exemple (Créer in index sur l'attribut 'Nom' pour la relation Patient)

```
CREATE INDEX INOM ON `Patient` (Nom ASC) ;
```

- **Supprimer un index.**

Syntaxe

```
DROP INDEX <nom_de_l'index>
```

Exemple (Supprimer l'index INOM de la relation Patient)

```
DROP INOM ;
```

3.2.3 Les vues

Une **vue** consiste en un ensemble d'enregistrements en provenance d'une ou plusieurs tables de la base de données. La clause WITH CHECK OPTION garantit que toute opération de mise à jour (INSERT, UPDATE, DELETE) exécutée sur la vue sera contrôlée pour voir si le changement est conforme à la définition de la vue. Un changement non conforme sera rejeté. Si cette clause n'est pas indiquée, les changements pourront être acceptés par les relations sur lesquelles est définie la vue.

- **Crée une vue.**

Syntaxe

CREATE VIEW <nom_de_la_vue> AS <commande_SELECT> [WITH CHECK OPTION]

Exemple (Créer la vue « Patient_Tlemcen » de la relation Patient)

CREATE VIEW 'Patient_Tlemcen' AS SELECT * FROM Patient WHERE Adresse='Tlemcen'.

- **Supprimer une vue.** Contrairement à la commande DROP TABLE, les données incluses dans la vue ne seront pas détruites. Seule la définition de la vue est supprimée du catalogue.

Syntaxe

```
DROP VIEW <nom_de_la_vue>
```

Exemple (Supprimer la vue 'Patient_Tlemcen')

```
DROP VIEW 'Patient_Tlemcen' ;
```

3.3 Le langage de manipulation des données (LMD)

Une requête se présente généralement sous la forme :

" SELECT ... FROM ... WHERE " :

- la clause **SELECT** exprime le résultat attendu sous la forme d'une liste d'attributs auxquels il est possible d'appliquer différents opérateurs et fonctions.

- La clause **FROM** liste les relations utilisées pour évaluer les requêtes;
- La clause **WHERE** qui est facultative énonce une condition que doivent respecter les enregistrements sélectionnés.

Syntaxe

```
SELECT [DISTINCT] { * | <nom_de_relation>,<nom_d'attribut> [alias] | <nom_d'attribut>[alias] [, ...]

FROM [<nom_d'utilisateur>]<nom_de_relation> [alias] [, ...]
[WHERE <condition>]

[GROUP BY <nom_d'attribut> [, ...] [HAVING <condition>] ]

[ {UNION | INTERSECT | MINUS [ALL]} <commande_SELECT>

[ ORDER BY {<nom_d'attribut> | <numéro_de_colonne>} [{ASC | DESC}] [, ... ]
```

3.3.1 Lien entre algèbre relationnelle et SQL

Soient les schémas relationnels R1, R2, R3 et R4 définis comme suit :

R1 (A :D1, B :D2)
R2 (C :D1, D :D2)
R3 (A :D1, E :D3)
R4 (B :D2)

Le tableau 3.1 donne le lien entre algebra relationnelle et SQL

Opération	**Expression algébrique**	**Expression SQL équivalente**
Projection	π A (R1)	SELECT A FROM R1
Sélection	σ<condition>(R1)	SELECT * FROM R1 WHERE <condition >
Produit cartésien	R1 X R2	SELECT * FROM R1, R2
Jointure	R1 ⋈ R3	SELECT * FROM R1, R3 WHERE R1.A = R3.A
Union	R1 U R2	SELECT * FROM R1 UNION SELECT * FROM R2
Intersection	R1 ∩ R2	SELECT * FROM R1, R2 WHERE R1.A = R2.C and R1.B = R2.D
Différence	R1 / R2	SELECT * FROM R1 WHERE not exists (SELECT * FROM R2 WHERE R2.C = R1.A and R2.D = R1.B)
Division	R1 ÷ R2	SELECT A FROM R1 GROUP BY A HAVING COUNT (distinct B) = SELECT count (distinct B) FROM R2)

Tableau 3.1: Lien entre algèbre relationnelle et SQL.

3.3.2 Les requêtes imbriquées

- **L'opérateur IN :** Permet de tester la présence d'une valeur particulière dans un ensemble.

Exemple (N° de téléphone des patients qui ont un médecin comme homonyme)

```
SELECT Telephone FROM 'Patient'
WHERE Nom IN (SELECT Nom FROM Médecin) ;
```

- **L'opérateur NOT IN : Permet de tester l'absence d'une valeur particulière dans un ensemble.**

Exemple (N° de téléphone des patients qui n'ont pas un médecin comme homonyme)

```
SELECT Telephone FROM 'Patient'
WHERE Nom NOT IN (SELECT Nom FROM Médecin) ;
```

- **L'opérateur ALL :** Compare chacune des valeurs de l'ensemble à une valeur particulière et retourne "VRAI" si la comparaison est évaluée pour chacun des éléments.

  Les comparateurs sont: <, <=, >, >=, =, != .

Exemple (Nom du médecin qui gagne le plus)

```
SELECT Nom FROM 'Médecin'
WHERE Salaire >= ALL (SELECT Salaire FROM Médecin);
```

- **L'opérateur ANY :** Compare chacune des valeurs de l'ensemble à une valeur particulière et retourne "VRAI" si la comparaison est évaluée à "VRAI" pour

au moins un des éléments.
Les comparateurs sont: <, <=, >, >=, =, != .

Exemple (Nom des patients qui possèdent un homonyme chez les médecins)

```
SELECT Nom FROM 'Patient'
WHERE Nom= ANY (SELECT Nom FROM Médecin);
```

- **L'opérateur EXISTS :** Retourne "VRAI" si une requête imbriquée retourne au moins une ligne.

Exemple (Liste des médecins qui ne sont pas patient)

```
SELECT Nom FROM 'Médecin'
WHERE NOT EXISTS (Patient) ;
```

3.3.3 Les prédicats

- **L'opérateur BETWEEN :** Teste l'appartenance d'une valeur à un intervalle.

Exemple (Nom et prénom des médecins qui gagne entre 100000 DA et 200000 DA)

```
SELECT Nom, Prénom FROM 'Médecin'
WHERE Salaire BETWEEN 100000 and 200000 ;
```

- **L'opérateur LIKE :** Teste l'appartenance d'une valeur à un intervalle.

Exemple (Nom des patients qui habitent Tlemcen)

```
SELECT Nom FROM 'Patient'
```

WHERE Ville **LIKE** '13---' **OR** Ville **LIKE** '%mcen%';

- **L'opérateur IS NULL :** Permet de tester si un champ a été affecté.

Exemple (Liste des médecins qui n'ont pas le téléphone)

SELECT Nom 'Patient'
WHERE Telephone IS NULL;

3.3.4 Les clauses

- **La clause GROUP BY :** Application de fonction agrégats à des collections d'enregistrements reliées sémantiquement.

Exemple (Nombre de patient de chaque ville)

SELECT Ville, count(*) FROM 'Patient'
GROUP BY Ville;

- **La clause HAVING :** Cette clause ne s'emploie qu'avec un "GROUP BY". Elle exprime une condition sur le groupe d'enregistrement associé à chaque valeur du groupage.

Exemple (Nombre de consultation effectuée avant le 31 décembre par patient)

SELECT Num_Patient, count(*) FROM 'Consultatin'
GROUP BY Num_Patient
HAVING Date_Consultation <= '31/12/2014';

- **La clause ORDER BY :** Cette clause permet l'ordonnancement du résultat avant l'affichage.

Exemple (Liste des salaires des médecins classés par ordre décroissant)

```
SELECT Salaire * 12 FROM 'Medecin'
ORDER BY Salaire DESC ;
```

- **La clause DISTINCT :** Cette clause Elimine les doublons avant d'utiliser une fonction agrégat.

Exemple (Liste de toutes les villes où habite au moins un patient)

```
SELECT DISTINCT Ville FROM 'Patient' ;
```

3.3.5 Les fonctions agrégats

Ces fonctions ne peuvent être utilisées que dans une clause SELECT ou dans une clause HAVING.

- **La fonction COUNT :** Compte les lignes sélectionnées.

Syntaxe

```
COUNT (Expression)
```

Exemple (Le nombre de patients)

```
SELECT COUNT(*) FROM 'Patient' ;
```

- **La fonction SUM :** Additionne les valeurs de type numérique.

Syntaxe

```
SUM (Expression)
```

Exemple (Somme des salaires des médecins dont le prénom est "Mohammed")

```
SELECT SUM(Salaire) FROM 'Medecin'
WHERE Prénom='Mohammed';
```

- **La fonction MIN :** Retourne la valeur minimale d'une colonne de type caractère ou numérique.

Syntaxe

MIN (Expression)

Exemple (Le médecin qui a le salaire le moins élevé)

```
SELECT Nom, MIN(Salaire) FROM 'Medecin' ;
```

- **La fonction MAX :** Retourne la valeur maximale d'une colonne de type caractère ou numérique.

Syntaxe

MAX (Expression)

Exemple (Le médecin qui a le salaire le plus élevé)

```
SELECT Nom, MAX(Salaire) FROM 'Medecin' ;
```

- **La fonction AVG :** Retourne la valeur moyenne d'une colonne de type numérique.

Syntaxe

AVG (Expression)

Exemple (Moyenne des salaires des médecins)

```
SELECT AVG(Salaire) FROM 'Medecin' ;
```

3.3.6 Instructions de mises à jour

- **Insertion de données dans une relation :** La commande **INSERT** permet d'ajouter des enregistrements à une relation dont le schéma a

été préalablement défini soit un enregistrement à la fois soit en utilisant une expression de sélection.

Syntaxe

INSERT INTO <nom_de_relation >
[(<liste_d'attributs>)]
VALUES (<liste_de_valeurs >)
Ou INSERT INTO <nom_de_relation>
[(<liste_d'attributs>)] <expression_de_sélection>

Exemple (Insertion d'un nouveau «enregistrement» patient)

```
INSERT INTO `patient` ( `Num_Patient` , `Nom` , `Prenom` , `Ville` , `EMAIL` , `Telephone` )
VALUES ('12', 'Omar', 'Ahmed', 'Tlemcen', 'omar.ahme@mail.univ-tlemcen.dz', '213 06 22 13 44');
```

- **Suppression de données dans une relation :** La commande **DELETE** permet de supprimer des enregistrements d'une relation. Cette opération n'affecte pas le schéma de la relation et ceci même dans le cas où la commande a pour effet de supprimer tous les enregistrements de la relation.

Syntaxe
DELETE FROM < nom_de_relation > [WHERE < expression_de_selection >]

Exemple 1 (Suppression de tous les enregistrements de la relation Patient)

```
DELETE * FROM Patient ;
```

Exemple 2 (Suppression du patient Num = 123)

```
DELETE FROM Patient WHERE Num_Patient = 123;
```

Remarque

Nous allons nous intéresser à 2 tables dont le schéma est le suivant :

```
CREATE TABLE `Patient` (
`Num_Patient` INT NOT NULL ,
`Nom` VARCHAR( 25 ) NOT NULL ,
`Prenom` VARCHAR( 25 ) NOT NULL ,
`Ville` VARCHAR( 45 ) NOT NULL
PRIMARY KEY (Num_Patient)) ;
```

et

```
CREATE TABLE Consultation (
id int(11) NOT NULL AUTO_INCREMENT,
Id_Patient int(11) NOT NULL,
Date_Consultation DATE,
Observation varchar(20),
PRIMARY KEY (id),
CONSTRAINT fk_Patient FOREIGN KEY (Id_Patient)
REFERENCES Patient (Num_Patient))
```

La suppression des enregistrements peut poser des problèmes de respect d'intégrité référentielle. Par exemple, si l'on souhaite supprimer un patient, la contrainte placée dans la définition de la table Consultation " CONSTRAINT fk_Patient FOREIGN KEY (Id_Patient) REFERENCES Patient (Num_Patient) " sera violée. Pour prendre en compte un tel cas, une clause " ON DELETE " peut être ajoutée en même temps que la définition de la contrainte d'intégrité référentielle.

ON DELETE {RESTRICT | SET DEFAULT | SET NULL | CASCADE}

Quatre options sont disponibles :

ON DELETE RESTRICT (ou NO ACTION selon le SGBD)

Permet d'interdire la suppression d'un enregistrement référencé par un enregistrement d'une autre relation.

ON DELETE SET NULL

Affecte la valeur NULL à la clé étrangère.

ON DELETE SET DEFAULT

Affecte la valeur par défaut (selon le type de données) à la clé étrangère.

ON DELETE CASCADE

Indique au SGBD qu'il est possible de détruire la clé étrangère en détruisant toutes les clés étrangères avec la même valeur.

Quand on n'indique pas de clause particulière dans la définition de la clé étrangère, cela équivaut à indiquer la clause NO ACTION (ON DELETE NO ACTION ON UPDATE NO ACTION). Cette action interdit la modification d'une ligne de la table Patient si cette ligne est en liaison avec la table Consultation par la clé étrangère. Le mot-clé RESTRICT est équivalent à NO ACTION.

- **Modification des données dans une relation :** La commande **UPDATE** permet de modifier un ou plusieurs individus.

Syntaxe

```
UPDATE <nom_de_relation> SET < champ >
```

Exemple (Modification du nom de « Mohammed »)

```
UPDATE Patient SET Nom="Ali" WHERE Nom="Mohammed";
```

3.4 Le langage de contrôle des données

Cette partie du langage SQL comporte deux commandes, la commande GRANT pour donner des droits à des utilisateurs et la commande REVOKE pour supprimer des droits préalablement accordés. Nous supposons ici que le SGBD permet de créer des utilisateurs.

3.4.1 Attribution de droits

La commande GRANT permet d'attribuer des privilèges. Les privilèges sont, pour un ou plusieurs utilisateurs la possibilité d'utiliser certains objets et parmi ces objets, certains ordres SQL. Les privilèges les plus courants sont la permission d'insérer (INSERT), de supprimer (DELETE) et la permission de sélectionner (SELECT) des enregistrements dans une relation donnée. Un utilisateur ne peut exécuter que les commandes SQL pour lesquelles les droits lui ont été explicitement attribués ou qu'il possède implicitement car l'objet accédé a été créé par lui-même. Les privilèges sur un objet (TABLE, VIEW) sont distribués soit pour un utilisateur donné, soit pour tous les utilisateurs.

Syntaxe

```
GRANT < privilège > [, ...] ON [TABLE]
<nom_de_relation>
TO {<nom_d'utilisateur > [, ...] | PUBLIC}
```

Exemple 1 (Toute personne ayant le droit d'accéder à la base de données peut accéder à la relation Patient)

```
GRANT SELECT ON 'Patient' TO PUBLIC
```

Exemple 2 (Autoriser l'utilisateur « Mohammed » à exécuter la commande SELECT sur la relation Patient)

GRANT SELECT ON 'Patient' TO 'Mohammed'

3.4.2 Suppression de droits

La commande REVOKE permet de supprimer des privilèges.

Syntaxe

```
REVOKE <privilège > [, ...] ON [TABLE]
<nom_de_relation>
FROM {<nom_d'utilisateur> [, ...] | PUBLIC}
```

Exemple (Supprimer l'autorisation de l'utilisateur « Mohammed » à exécuter la commande INSERT sur la relation Patient)

REVOKE INSERT ON 'Patient' FROM 'Mohammed'

3.5 Exercices avec solutions

3.5.1 Exercice 1

Soit la relation R de schéma :

R (NCom, NC, NomC, DateC, QteC, Montant)

Avec : NCom (Numéro de Commande), NC (Numéro du client), NomC (Nom du client), DateC (Date de commande), QteC (Quantité commandée)

1) On désire obtenir les noms des clients ainsi que leur n° commande.
2) On désire obtenir les noms des clients ayant déposé des commandes le 20/03/80.

3) On désire obtenir les noms des clients ayant commandé une quantité > 15 et dont le montant est > 2000.

4) On désire obtenir la liste des clients classés par ordre alphabétique ainsi que leurs numéros.

Question : effectuer les différentes requêtes en langage SQL.

Solution

1) SELECT NomC, NCom
   FROM R
2) SELECT NomC
   FROM R
   WHERE DateC = '20/03/80'
3) SELECT NomC
   FROM R
   WHERE QteC > 15 AND Montant > 2000
4) SELECT NomC, NC
   FROM R
   ORDER By NomC ASC

3.5.2 Exercice 2

Soit le schéma des relations suivant :

Command (NCom, DateC, NC, NomC, AdrC)
LIG-CDE (NCom, Code_Produit, QteC)
PRODUIT (Code_Produit, Unité_Mesure, Designation)

Avec : NCom (Numéro de Commande), DateC (Date de commande), NC (Numéro du client), NomC (Nom du client), AdrC (Adresse Client), QteC (Quantité commandée).

Question : Donner la requête SQL permettant d'obtenir la commande (NCom, DateC, NC) dans laquelle apparait le produit " Sucre".

Solution

SELECT NCom, DateC, NC

FROM Command, LIG-CDE, PRODUIT

WHERE Command.NCom = LIG-CDE.NCom

AND LIG-CDE.Code_Produit = PRODUIT.Code_Produit

AND PRODUIT. Designation = ″Sucre″

3.5.3 Exercice 3

On considère le schéma relationnel suivant qui modélise une application sur la gestion de livres dans une bibliothèque[4] :

- LIVRE (CodeOuv, Titre, Genre, Editeur, Collection)

Avec : CodeOuv (code de l'ouvrage), Genre (par exemple Policier, Roman, Informatique), Editeur (par exemple Ibn Khadoun), Collection (par exemple livre de poche). Ces informations sont générales et pour un enregistrement de la relation LIVRE, on aura n (n > 1) enregistrements dans la relation E_LIVRE

[4] Cet exercice est repris (avec quelques modifications) du site web: https://www.cours-gratuit.com/langage-sql

correspondant aux exemplaires de ce livre possédés par la bibliothèque.

- E_LIVRE (CodeOuv, NumEx, DateAchat, Etat)

Cette relation contient un enregistrement pour chaque exemplaire de livre possédé par la bibliothèque. Chaque exemplaire est identifié par son code (CodeOuv) et un numéro d'exemplaire (NumEx). On trouve également la date d'achat (DateAchat) et l'état du livre (par exemple Neuf ou Abimé).

- AUTEUR (CodeOuv, Identité)

Chaque enregistrement de cette relation correspond à l'un des auteurs d'un ouvrage. L'attribut identité peut avoir pour valeur un nom de personne (par exemple Abderrahim Mohammed El Amine).

- ABONNE (NumAbo, Nom, Prénom, Rue, Ville, CodeP, Téléphone)

Cette relation représente les abonnés de la bibliothèque : NumAbo identifie tout abonné de manière individuelle, le nom (Nom) et le prénom (Prénom) de l'abonné, son adresse (Rue, Ville, CodeP), son téléphone (Téléphone).

- PRET (CodeOuv, NumEx, NumAbo, DatePrêt, DateRestitution)

Cette relation contient un enregistrement par prêt effectué. Pour chaque prêt, on trouve l'identifiant du livre (CodeOuv et NumEx), le numéro de l'abonné effectuant le prêt (NumAbo), la date du prêt (DatePrêt) et la date de restitution(DateRestitution). Cette relation ne contient des informations que pour les prêts en cours c'est à dire pour les emprunts non encore rendus.

- PERSONNEL (NumEmp, Nom, Prénom, Adresse, Fonction, Salaire)

Cette relation contient un enregistrement par employé. Chaque employé est identifié par un numéro (NumEmp). Pour chaque employé, la relation donne son nom, son prénom, son adresse, sa fonction et son salaire mensuel.

a) Requêtes simples

Q1 : Quel est le contenu de la relation LIVRE ?

Solution

```
SELECT *
FROM LIVRE
```

Q2 : Quels sont les titres des romans édités par Ibn Khaldoun ?

Solution

```
SELECT Titre
FROM LIVRE
WHERE Editeur = 'Ibn Khaldoun' and Genre = 'Roman'
```

Q3 : Quelle est l'identité des auteurs qui ont écrit des romans ?

Solution

```
SELECT A1.Identité
FROM  LIVRE L, AUTEUR A1
WHERE L.CodeOuv = A1.CodeOuv
AND L.Genre = 'Roman'
```

b) Requêtes avec des clauses

Q4 : Quels sont les différents genres de livre proposés ?

Solution

```
SELECT distinct Genre
FROM LIVRE
```

La clause **distinct** permet de supprimer les doublons.

Q5 : Quel est le salaire annuel des membres du personnel gagnant plus de 500 000 DA en ordonnant le résultat par salaire descendant et le nom croissant ?

Solution

```
SELECT Nom, Prénom, Salaire * 12
FROM PERSONNEL
WHERE Salaire * 12 > 500 000
ORDER BY Salaire DESC, Nom ASC
```

Q6 : Donnez le nombre de prêts en cours pour chaque famille en considérant qu'une famille regroupe des personnes de même nom et possédant le même numéro de téléphone ?

Solution

```
SELECT Nom, Téléphone, count(*)
FROM ABONNE A, PRET P
WHERE A.NumAbo = P.NumAbo
GROUP BY Nom, Téléphone
```

Q7 : Quel est le code du livre dans la bibliothèque qui possède le plus grand nombre d'exemplaires?

Solution

```
SELECT CodeOuv
FROM E_LIVRE
GROUP BY CodeOuv
HAVING COUNT(*) = ( SELECT max(count(*))
FROM E_LIVRE
GROUP BY CodeOuv )
```

c) Requêtes avec des prédicats

Q8 : Quels sont les éditeurs pour lesquels l'attribut Collection n'a pas été renseigné ?

Solution

```
SELECT Editeur
FROM LIVRE
WHERE Collection IS NULL
```

Q9 : Quels sont les abonnés dont le nom contient la chaîne " ABD " et habitant la ville de Tlemcen ?

Solution

```
SELECT *
FROM ABONNE
WHERE Nom = '%ABD%' and CodeP ='13---'
```

- Le caractère de remplacement **%** indique la possibilité d'avoir 0 ou plusieurs caractères quelconques.
- Le caractère spécial **-** indique l'obligation d'avoir exactement un caractère quel que soit sa valeur.

d) Requêtes avec des agrégats

Q10 : Quel est le nombre de prêts en cours ?

Solution

```
SELECT count(*)
FROM PRET
WHERE PRET.DateRestitution=NULL
```

Q11 : Quels sont les salaires minimum, maximum et moyen des employés exerçant une fonction de bibliothécaire ?

Solution

```
SELECT min(Salaire), max(Salaire), avg(Salaire)
FROM PERSONNEL
WHERE Fonction = 'bibliothécaire'
```

Q12 : Quel est le nombre de genres de livres différents ?

Solution

```
SELECT COUNT(DISTINCT Genre)
FROM LIVRE
```

Q13 : Quel est le nombre de livres achetés en 2006 ?

Solution

```
SELECT count(*)
FROM E_LIVRE
WHERE DateAchat >= '01-JAN-2006' and DateAchat <= '31-DEC-2006'
```

e) Requêtes avec des opérateurs numériques

Q15 : Quel est le salaire annuel des membres du personnel gagnant plus de 150 000 DA ?

Solution

```
SELECT Nom, Prénom, Salaire * 12
FROM PERSONNEL
WHERE Salaire * 12 > 150 000
```

f) Requêtes avec des jointures et opérations ensemblistes

Q16 : Quel est le nom, le prénom et l'adresse des abonnés ayant emprunté un livre le ' 12-JAN-2006 ' ?

Solution

```
SELECT Nom, Prénom, Rue, Ville, CodeP
FROM ABONNE A, PRET P, LIVRE L
WHERE A.NumAbo = P.NumAbo and P.CodeOuv = L.CodeOuv
and P.DatePrêt = '12-JAN-2006 '
```

Q17 : Quels sont les titres des livres actuellement empruntés par Ahmed Ali ?

Solution

```
SELECT Titre
FROM ABONNE A, PRET P, LIVRE L
WHERE A.NumAbo = P.NumAbo and P.CodeOuv = L.CodeOuv
and A.Nom = 'Ahmed ' and A.Prénom = ' Ali '
```

3.6 Conclusion

Dans ce chapitre nous avons étudié le langage SQL (Structured Query Language) qui peut être considéré comme le langage d'accès normalisé aux bases de données. Il est aujourd'hui supporté par la plupart des produits commerciaux que ce soit par les systèmes de gestion de bases de données micro tel que « Access » ou par les produits plus professionnels tels que « Oracle ». Il a fait l'objet de plusieurs **normes ANSI/ISO** dont la plus répandue aujourd'hui est

la norme SQL2 qui a été définie en 1992. Nous avons décrit les principaux aspects de cette norme.

4 Conclusion générale

Ce polycopié a introduit la notion de conception et d'implémentation d'une BD en utilisant le modèle relationnel avec un langage algébrique, SQL.

Autrement, nous distinguons pour le modèle relationnel deux langages :

1. Algébrique : SQL
2. Prédicatif : Calcul relationnel sur triples (QUEL) / Calcul relationnel sur domaine (QBE)

Le modèle relationnel peut être introduit par :

1. Les structures de données sont simples et se construisent à partir de la théorie des ensembles.

2. Une algèbre relationnelle permettant la définition, la recherche et la mise à jour des données

3. Un ensemble de contraintes d'intégrité sémantique exemple incité de la clé,….

Avantage du modèle relationel

- Langage non navigationnel,
- Simple d'utilisation,
- Indépendance entre le niveau physique et le niveau logique,

Limite des SGBD relationnels

Ne permet pas la gestion :

- Des objets complexes et dynamiques,
- Des connaissances,
- Des données réparties,

D'où la naissance de la 3^{éme} génération des SGBD : orientée objet, détective, repartie,….

5 Bibliographie

1. Georges Gardarin, Bases de données objet et relationnel. Eyrolles, 1999.

2. Georges Gardarin, Bases de données : Les systèmes et leurs langages, Eyrolles, 1985.

3. André Flory, Bases de données : Conception et réalisation, Economica, 1987.

4. Nacer Boudjlida, Bases de données et systèmes d'informations : Le modèle relationnel : langages, systèmes et méthodes, Dunod, 1999.

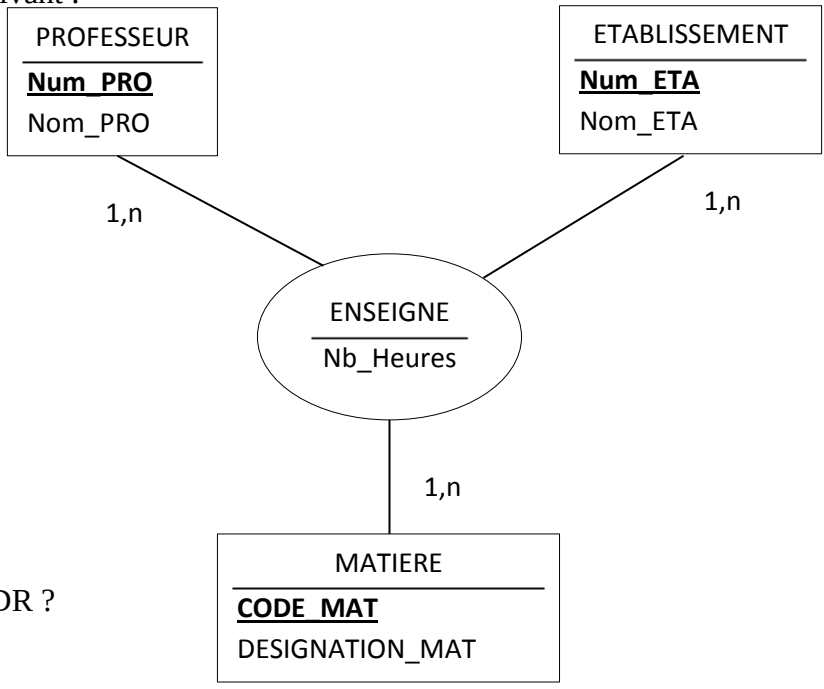
5. Jean-Luc Hainaut, Bases de données et modèles de calcul : outils et méthodes pour l'utilisateur, Dunod, 2000.

6. Raghu Ramakrishnan, Johannes Gehrke. Database Management Systems. 2nd edition. Mc Graw-Hill,1999.

ANNEXE : Exercices avec solutions

Exercice 1

Soit le MCD suivant :



Trouver le MLDR ?

Solution exercice 1

PROFESSEUR (**Num_PRO**, Nom_PRO)
ETABLISSEMENT (**Num_ETA**, Nom_ETA)
MATIERE (**Code_MAT**, Désignation_MAT)
ENSEIGNE(#Num_PRO, #Num_ETA , #Code_MAT, Nb_Heures)

Exercice 2

NOTION DE RETROCONCEPTION

La rétro-conception est l'opération qui consiste à retrouver le MCD à partir d'un schéma relationnel. L'intérêt peut être de faire évoluer une base de données existante. En effet, il est plus facile d'appliquer et de vérifier des règles de gestion sur un MCD que sur un modèle relationnel.

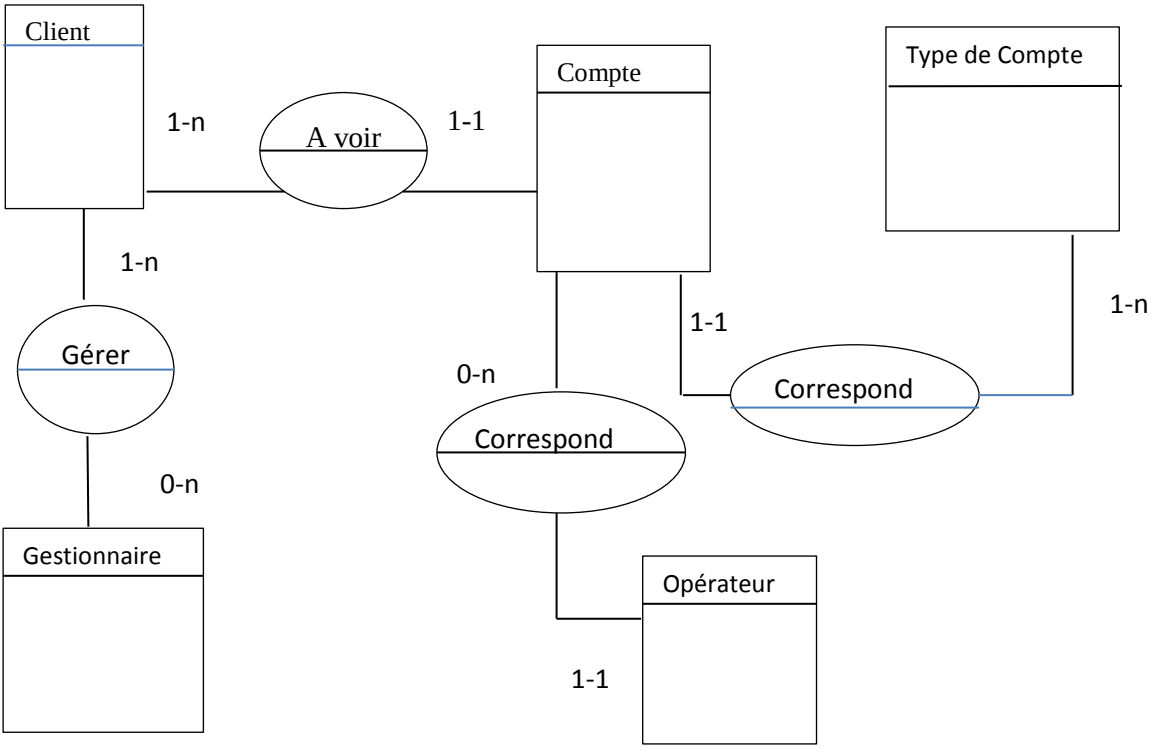
Réaliser le MCD correspondant au schéma relationnel suivant :

Soit le schéma relationnel suivant qui correspond à une base permettant la gestion des comptes clients d'une banque :

Client (NumClient, NomClient, PrenomClient, AdrClient)
Compte (NumCpt, DateOuvertureCpt, #NumClient, #CodeTypeCpt)
TypeCompte (CodeTypeCpt, LibTypeCpt)
Opération(NumOpe, SensOpe, MontantOpe, #NumCpt)
Gestionnaire (NumGest, NomGest)
Gérer (#NumGest, #NumCli)

Avec les règles de gestion suivantes :
RG 1 :Un client est enregistré dès lors qu'il possède un compte ; on lui fournit alors les
références des personnes chargées de gérer ses comptes.
RG 2 : Il se peut qu'un gestionnaire nouvel arrivant ne soit pas encore affecté à la gestion de
comptes clients.
RG 3 : On ne gère que les types de comptes pour lesquels un compte a été ouvert.

Solution exercice 2



Exercice 3

Etant donné le MLDR suivant :

PERSONNE(Id_Personne, Nom, Prénom, #Id_Département, Sexe, Salaire)
DEPARTEMENT (Id_Département, Nom_Département, Bâtiment, Budget)

1) Trouvez le MCD (Entité/Association) relatif à ce modèle.

2) Ecrivez des requêtes en algèbre relationnelle permettant de :

a) Trouver le nom, le département et le salaire de tous les employés.

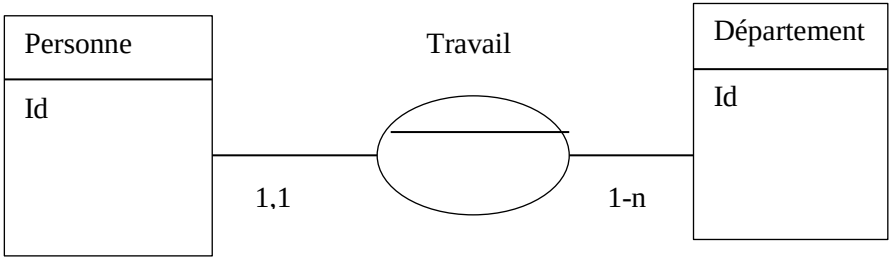
b) Trouver le budget du département où travaille l'employé numéro 5 (ie : Id_Personne=5).

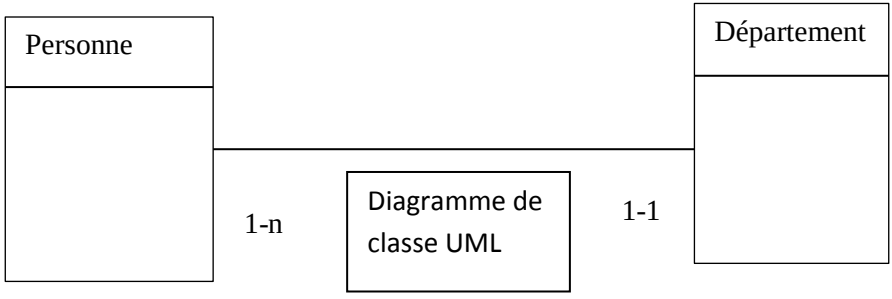
c) Trouver le nom et prénom de chaque employé, son département et le bâtiment où se trouve son département.

d) Trouver le nom et le département de tous les employés ayant un salaire supérieur à 15000 DA.

Solution exercice 3

1) Le MCD (Entité/Association) relatif à ce modèle





2) Requêtes en algèbre relationnelle

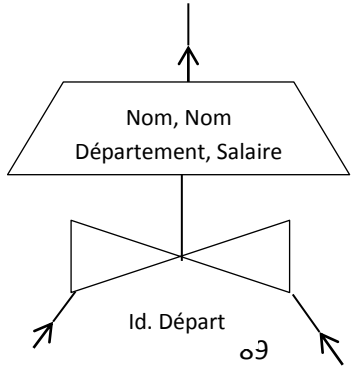
a) Trouver le nom, le département et le salaire de tous les employés.

Select Nom, Nom Département, Salaire

From Personne, Département

Where Personne. Id_Département =

Département. Id_ Département



R′1 = Personne ⋈ Département
Id_Départ

R1 = π Nom, Nom Département, Salaire (R′1)

b) Trouver le budget du département où travaille l'employé numéro 5 (ie : Id_Personne=5).

Select Budget

From Personne, Département

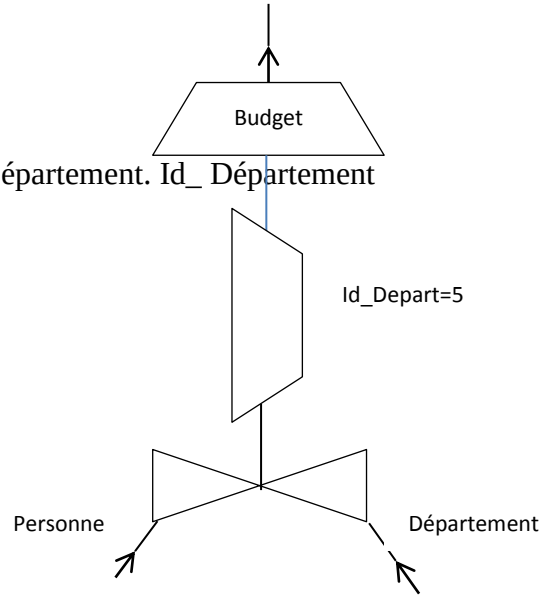
Where Personne. Id_Département, = Département. Id_ Département

And Personne. Id_ Personne = 5

R′2 = Personne ⋈ Département
Id_Départ

R″2 = σ Id_ Personne = 5 (R′2)

R2 = π Budget (R″2)



c) Trouver le nom et prénom de chaque employé, son département et le bâtiment où se trouve son département.

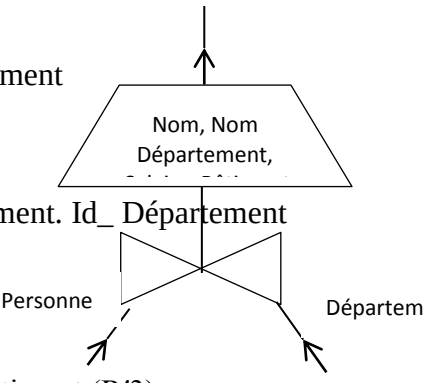
Select Nom, Nom Département, Salaire, Bâtiment

From Personne, Département

Where Personne. Id_Département = Département. Id_ Département

R′3 = Personne ⋈ Département
Id_Départ

R3 = π Nom, Nom Département, Salaire, Bâtiment (R′3)



d) Trouver le nom et le département de tous les employés ayant un salaire supérieur à 15000 DA.

Select Nom, Nom Département

From Personne, Département

Where Personne. Id_Département =

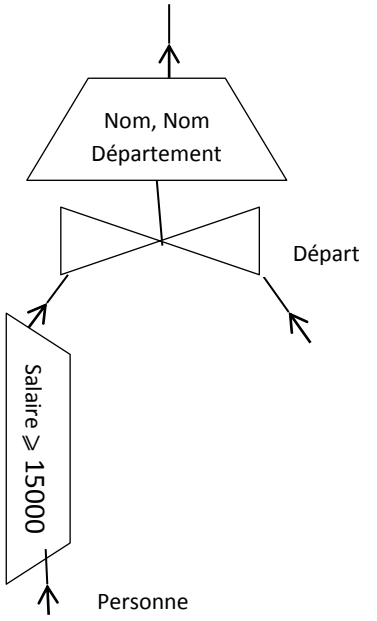
Département. Id_ Département

And Personne. Salaire ⩾ 15000

R′4 = σ Salaire ⩾ 15000 (Personne)

R″4 = Personne ⋈ R′4

R4 = π Nom, Nom Département



Exercice 4

Etant donné le MLDR suivant :

Auteur (Id_Auteur, Nom_Auteur, Prénom_Auteur, Date_Naissance_Auteur, Adresse_Auteur)

Livre (Id_Livre, Titre, Année, Résumé, Prix, #Id_Editeur)

Rédiger (#Id_Auteur, #Id_Livre)

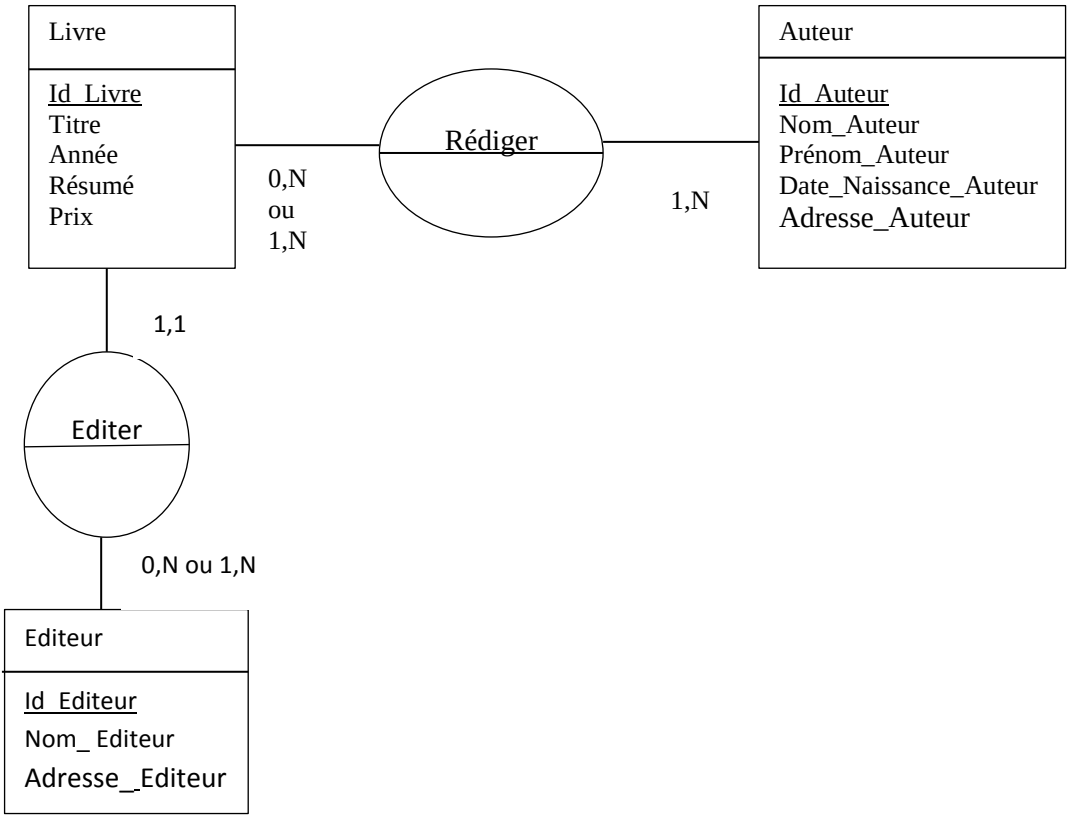
Editeur (Id_Editeur, Nom_Editeur, Adresse_Editeur)

NB : Un auteur rédige au moins un livre

1) Trouvez le MCD (Entité/Association) relatif à ce modèle.
2) Ecrivez des requêtes en algèbre relationnelle (ou en SQL) permettant de :
   a) Trouver le nombre des auteurs de la ville de Tlemcen.
   b) Trouver les noms des éditeurs de la ville de Tlemcen.
   c) Trouver le nom, le prénom et la date de naissance de tous les auteurs de la ville de Tlemcen.
   d) Trouver le nom et l'adresse des éditeurs des livres rédigés par l'auteur numéro 5 (ie : Id_Auteur=5).
   e) Trouver le nombre de livres rédigés par l'auteur « Abderrahim ».
   f) Trouver le titre et le résumé de tous les livres rédigés par l'auteur « Abderrahim ».
   g) Trouver le nom de l'auteur, le titre du livre et le nom de l'éditeur du livre le plus cher.

Solution exercice 4

1) Le MCD (Entité/Association) relatif au modèle.



2) Les requêtes en algèbre relationnelle (ou en SQL)

a) Trouver le nombre des auteurs de la ville de Tlemcen.

```
Select count (*)
From Auteur
Where Auteur = " Tlemcen "
```

b) Trouver les noms des éditeurs de la ville de Tlemcen.

```
Select Nom_Editeur
From Editeur
Where Adresse_ Editeur =  " Tlemcen "
```

c) Trouver le nom, le prénom et la date de naissance de tous les auteurs de la ville de Tlemcen.

```
Select Nom_ Auteur, Prénom_Auteur, Date_Naissance_Auteur
```

From Auteur
Where Adresse_Auteur = " Tlemcen "

d) Trouver le nom et l'adresse des éditeurs des livres rédigés par l'auteur numéro 5 (ie : Id_Auteur=5).

Select Nom_Editeur, Adresse_ Editeur
From Editeur, Rédiger, Livre
Where Editeur. Id_ Editeur = Livre. Id_ Editeur
And Livre. Id_ Livre. = Rédiger, Id_Livre
And Rédiger. Id_Auteur = 5

e) Trouver le nombre de livres rédigés par l'auteur « Abderrahim ».

Select count (*)
From Rédiger, Auteur
Where Rédiger. Id_ Auteur = Auteur. Id_ Auteur
And Auteur. Nom_ Auteur = "Adresse"

f) Trouver le titre et le résumé de tous les livres rédigés par l'auteur « Abderrahim ».

Select Titre, Résumé
From Rédiger, Auteur
Where Rédiger. Id_ Auteur = Auteur. Id_ Auteur
And Rédiger.= Id_ Livre. = Livre., Id_Livre
And Auteur. Nom_ Auteur = "Adresse"

g) Trouver le nom de l'auteur, le titre du livre et le nom de l'éditeur du livre le plus cher.

Select Nom_ Auteur, Titre_ Livre, Nom_Editeur, Max (Prix)
From Auteur, Livre, Editeur, Rédiger
Where Auteur. Id_ Auteur = Rédiger. Id_ Auteur
And Rédiger.= Id_ Livre. = Livre., Id_Livre
And Livre. Id_ Editeur = Editeur. Id_ Editeur

Exercice 5

Soit une BDD composée de trois relations :
Command(*Num_Command*, Date_Command, Num_Client, Nom_Client, Adresse_Client)
Lig_Cde(*Num_Command*, *Code_Prod*, Qte)
Produit(*Code_Prod*, UM, Designation)

1) Exprimez les requêtes suivantes en SQL.
   a) On désire obtenir les noms des clients ainsi que leur numéro de commande
   b) On désire obtenir les noms des clients ayant déposé des commandes le 17/09/2003
   c) On désire obtenir les noms des clients ayant commandé une quantité supérieur à 20
   d) On désire obtenir les commandes dans lesquelles apparaît le produit « Farine »
2) Donner la représentation de l'arbre syntaxique pour les requêtes précédentes.

Solution exercice 5

1) Exprimez les requêtes suivantes en SQL.

a) Select Nom_ Client, Num_Command
From Command

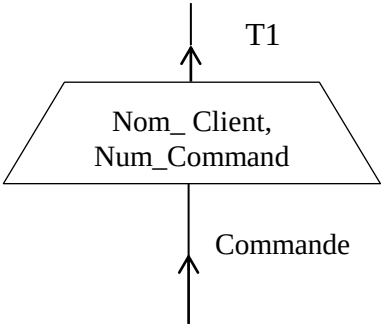
b) Select Nom_ Client
From Command
Where Command. Date_ Command = ″ 17/09/2003″

c) Select Nom_ Client
From Command, lig_cde
Where Command. Num_Command = lig_cde. Num_Command
And lig_cde . Quantité > 20

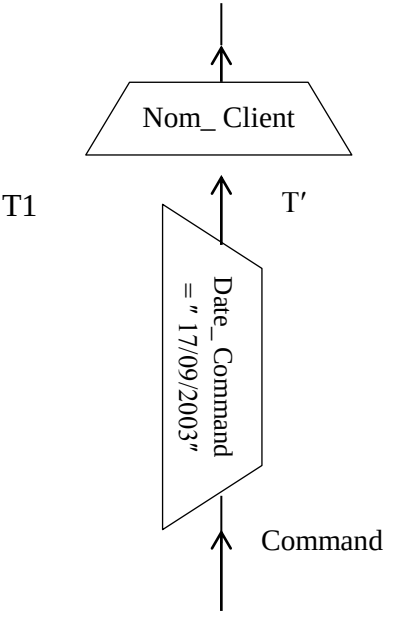
d) Select Num_Command
From lig_cde, Produit
Where lig_cde . cde_ Produit = Produit. cde_ Produit
And Produit . Désignation = ″ Farine ″

2) La représentation de l'arbre syntaxique pour les requêtes précédentes.

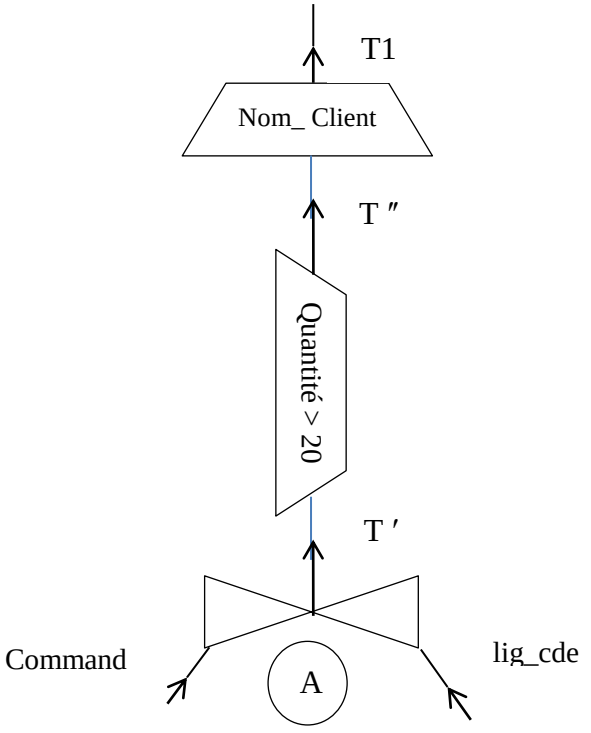
a)



b)

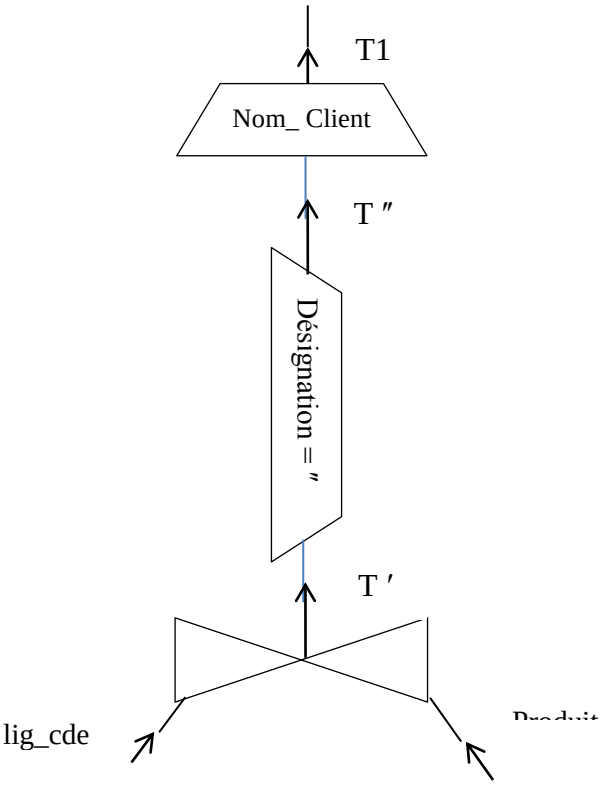


c)



A : Command. Num_Command = lig_cde. Num_Command

d)



B : lig_cde . cde_ Produit = Produit. cde_ Produit

Exercice 6

Soit le schéma suivant:

- Magasins (nom_mag, secteur, ville_mag)
- Clients (nom_client, rue, ville)
- Employés (nom_emp, nom_mag, équipe, salaire)
- Achats (nom_client, nom_mag, vendeur, montant, date, heure) (Vendeur = nom_emp)

1) Ecrire en SQL les requêtes suivantes

a) Nom des magasins du secteur 92.

b) Nom des clients habitant Tlemcen par ordre alphabétique.

c) Quels sont les clients qui ont acheté quelque chose au magasin de la ville d'Oran ?

2) Donner la représentation de l'arbre relationnel pour chaque question précédente.

Solution exercice 6

1) Les requêtes en SQL

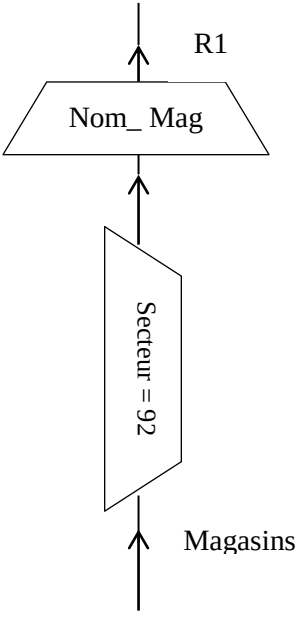
a)
```
SELECT Nom_ Mag
  FROM Magasins
 WHERE Secteur = 92
```

b)
```
SELECT Nom_ Client
  FROM Client
 WHERE Ville = ″ Tlemcen ″
 Order By Nom_ Client Descending
```

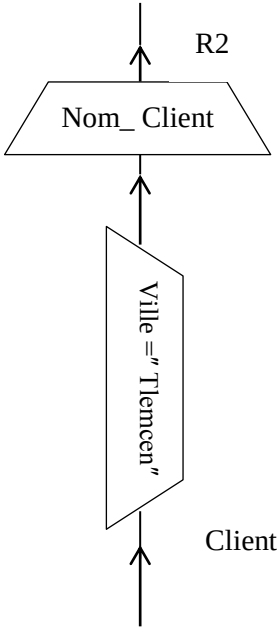
c)
```
SELECT Nom_ Client
  FROM Achats, Magasins
 WHERE Achats. Nom_Magasins = Magasins. Nom_Magasins
 AND Magasins. Ville_Mag = ″ Oran ″
```

2) La représentation de l'arbre relationnel pour chaque question précédente.

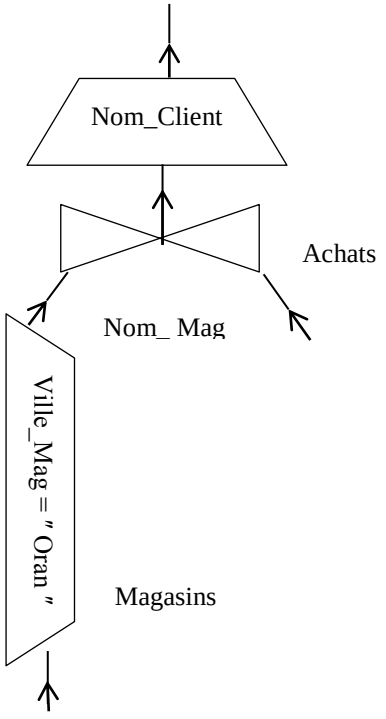
a)


R1
Nom_ Mag
Secteur = 92
Magasins

b)


R2
Nom_ Client
Ville =" Tlemcen"
Client

c)



Exercice 7

Soient les deux relations R et S (R représente les étudiants et S les notes obtenues) :

R	N°	Nom	DN	Ville
	1	Med	D1	V1
	2	Fatima	D2	V2
	3	Ali	D1	V2
	4	Omar	D3	V1

S	N°	EF1	EF2
	1	14	12
	2	15	13
	3	10	16
	5	11	10

DN : date de naissance. EF : Epreuve Finale

Effectuez les opérations suivantes :

T = π Nom, DN(R) ; T1 = π Ville (R) ; T2 = R $\bowtie$ S ;

T3 = R X S ; T4 = R - S ; T5 = π Nom, EF1(T2) ;

Solution exercice 7

T= π Nom, DN (R)

T3	Nom	DN
	Mohamed Fatima Ali Omar	D1 D2 D1 D3

T1 = π Ville (R)

T3	Ville
	V1 V2 V2 V1

T2 = R ⋈ S

T3	N°	Nom	DN	Ville	EF1	EF2
	1	Mohamed	D1	V1	14	12
	2	Fatima	D2	V2	15	13
	3	Ali	D1	V2	10	16

T3 = R X S

T3	N°	Nom	DN	Ville	N°	EF1	EF2
	1	Mohamed	D1	V1	1	14	12
	1	Mohamed	D1	V1	2	15	13
	1	Mohamed	D1	V1	3	10	16
	1	Mohamed	D1	V1	5	11	10
	2	Fatima	D2	V2	1	14	12
	2	Fatima	D2	V2	2	15	13
	2	Fatima	D2	V2	3	10	16
	2	Fatima	D2	V2	5	11	10
	3	Ali	D1	V2	1	14	12
	3	Ali	D1	V2	2	15	13
	3	Ali	D1	V2	3	10	16
	3	Ali	D1	V2	5	11	10
	4	Omar	D3	V1	1	14	12
	4	Omar	D3	V1	2	15	13
	4	Omar	D3	V1	3	10	16
	4	Omar	D3	V1	5	11	10

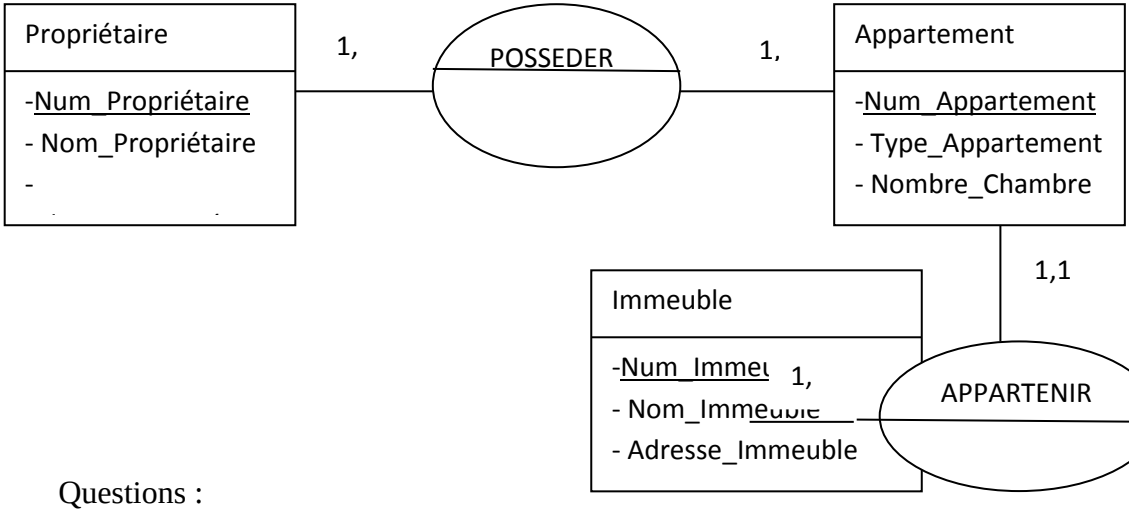
T4= R – S : Impossible

T5= π Nom, EF1 (T2)

T5	Nom	EF1
	Mohamed	14
	Fatima	15
	Ali	10

Exercice 8

Soit le MCD suivant :



Questions :

1- Etablir le MLD relationnel
2- Exprimer les questions suivantes en SQL :
   a) Donner les noms des propriétaires ayant au moins un appartement dont le nombre de chambre est égale à 3.
   b) Donner les noms des propriétaires ayant au moins un appartement dans un Immeuble situé à Tlemcen (Adresse_Immeuble= 'Tlemcen').
   c) Donner le nombre de propriétaires ayant au moins un appartement dans un Immeuble situé à Tlemcen (Adresse_Immeuble= 'Tlemcen').

Solution exercice 8

1) le MLD relationnel

Propriétaire (Num_ Propriétaire, Nom_ Propriétaire, Adresse_ Propriétaire)
Appartement (Num_ Appartement, Type_ Appartement, Nombre_Chambre, # Num_ Propriétaire , # Num_Immeuble)
Immeuble (Num_ Immeuble, Nom_ Immeuble, Adresse_ Immeuble)

2) Les requêtes en SQL

a) Select Nom_ Propriétaire

From Propriétaire, Appartement

Where Propriétaire. Num_ Propriétaire = Appartement . Num_ Propriétaire

And Appartement. Nombre_chambre = 3

b) Select Nom_ Propriétaire

From Propriétaire, Appartement, Immeuble

Where Propriétaire. Num = Num . Appartement

And Immeuble. Adresse _ Immeuble = ″ Tlemcen ″

c) Select count (*)

Select Nom_ Propriétaire

From Propriétaire, Appartement, Immeuble

Where Propriétaire. Num = Num . Appartement

And Immeuble. Adresse _ Immeuble = ″ Tlemcen ″

Exercice 9 (Algèbre relationnelle)

Soient les deux relations R et S :

R	A	B	C
	a1	b1	c1
	a1	b1	c3
	a2	b2	c5
	a5	b4	c2

S	C	D
	c1	d1
	c4	d2
	c5	d1

a) Effectuez les opérations suivantes :

T = R ⋈ S ; T1 = π A, C (T) ; T2 = π A (T1) ;

T3 = π A, C (R) ; T4 = T2 ⋈ T3 ; T5 = R - S ;

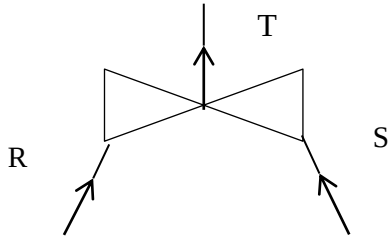
b) Donnez l'arbre relationnel des opérations T, T1, T2, T3, T4.

Solution exercice 9

Les opérations avec l'arbre relationnel

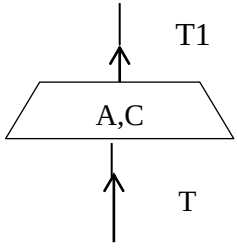
T = R ⋈ S

T	A	B	C	D
	a1 a2	b1 b2	c1 c2	d1 d2



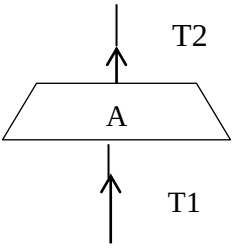
T1 = π A, C (T)

T1	A	C
	a1 a2	c1 c5



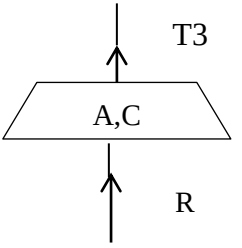
T2 = π A (T1)

T2	A
	a1
	a2



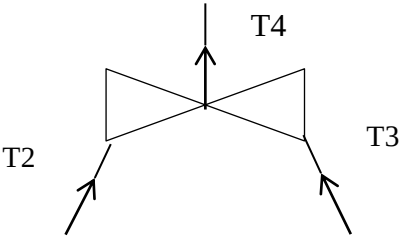
T3 = π A, C (R)

T3	A	C
	a1	c1
	a2	c2
	a2	c5
	a5	c2



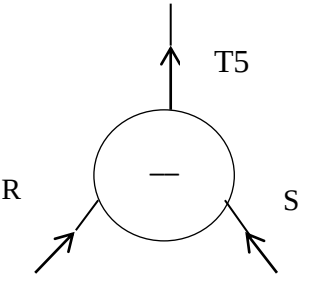
T 4= T2 $\bowtie$ T3

T4	A	C
	a1	c1
	a2	c3
	a2	c5



T 5 Impossible

T 5 = R – S



Exercice 10

Soit le schéma relationnel de la base de données du personnel d'une entreprise, ci-dessous.

Employé (**NOEMP**, Nom, Fonction, *MANAGER*, DateEmbauche, Salaire, Commission, *NODEPT*)

Département (**NODEPT**, Intitulé, Localisation)

NB : La clé étrangère MANAGER (supérieur hiérarchique) fait référence à un numéro d'employé (NOEMP) de la relation EMPLOYE.

Formuler en algèbre relationnelle les requêtes suivantes.

1. Nom des employés dont la fonction est vendeur et dont la commission est inférieure à 25 % du salaire.
2. Nom, salaire et commission des employés des départements localisés à Tlemcen.
3. Liste des employés avec leur nom et celui de leur supérieur hiérarchique.
4. Nom des employés de même fonction que Mohammed.
5. Nom des employés ne travaillant pas dans le même département que leur manager.

Solution exercice 10

Les requêtes en algèbre relationnelle

1)

$R11 = \sigma$ (fonction = ″vendeur″ et commission < (25 * salaire) /100)
(Employé)

$R1 = \pi$ Nom (R11)

2)

R11 = Employé $\bowtie$ Département
N° Dépt

R12 = σ Localisation = ″Tlemcen″ (R11)

R2 = π Nom, salaire, commission (R12)

3)

R11 = Employé $\bowtie$ Employé
N° Emp = Manager

R3 = π N° Employé/a, Employé. Nom, Employé/b. Nom (R11)

4)

R11 = σ Nom = ″Mohammed″ (Employé)

R12 = Employé $\bowtie$ R11
Fonction

R4 = π Nom (R12)

5)

R11 = Employé/a $\bowtie$ Employé/b
N° Emp = Manager

R12 = σ a. N° Emp $\neq$ b. N° Départ (R11)

R5 = π Nom (R12)