Base de Données

Rafik KHEMAKHEM

Objectif du cours : Ce cours permettra aux étudiants de concevoir et d'implémenter une base de données en passant par les différents niveaux : (conceptuel, logique et physique). Il aura l'occasion de bien assimiler l'apport des SGBD pour gérer les données. Ce cours de Base de Données est attribué aux étudiants de la ***Licence Fondamentale en Sciences de l'informatique - Parcours : Informatique et Multimédia*** (Semestre 3).

Volume Horaire Semestriel :

- 42 Heures de Cours
- 21 Heures de Travaux Dirigés

Etablissement : Institut **S**upérieur d'**I**nformatique et de **M**ultimédia de **G**abès

Table des matières

Chapitre

I

Introduction et Principaux Concepts des BD

I.1. Historique

Une base de données BD est un ensemble d'informations partagé par plusieurs utilisateurs. Ces informations sont interrogées et mises à jour par l'intermédiaire d'un logiciel. Comme exemple de bases étudiées on peut citer :

— Gestion des notes des étudiants
— Annuaire électronique
— Catalogue électronique d'une bibliothèque
— Etc ...

D'une manière générale, on peut considérer une Base de Données (BD) comme une grande quantité de données, servant pour les besoins d'une ou plusieurs applications, interrogeables et modifiables par un groupe d'utilisateurs travaillant en parallèle. Alors qu'un Système de Gestion de Bases de Données (SGBD), il peut être vu comme le logiciel qui prend en charge la structuration, le stockage, la mise à jour et la maintenance des données ; c'est, en fait, l'interface entre la base de données et les utilisateurs ou leurs programmes. Le modèle de données relationnel a été défini en 1970 par l'informaticien britannique Edgar F. Codd, et publié dans sa thèse A Relational Model of Data for Large Shared Data Banks. En 2010 le modèle de données relationnel est utilisé dans la grande majorité des bases de données.

— Jusqu'à 1960 : organisation classique en fichiers
— Fin 1960 : apparition des premiers SGBD, systèmes réseaux hiérarchiques
— Début 1970 : 2ème génération des SGBD, systèmes relationnels
— Début 1980 : 3ème génération des SGBD, systèmes orientés objets

I.2. Concepts de base

I.2.1. Pourquoi une base de données ?

La prise en compte de masse d'informations importantes dans des environnements riches soulève plusieurs difficultés :

• Maîtrise de la représentation de données complexes : Comment représenter des informations très diverses, très complexes, relevant de différents domaines (à l'intérieur de l'entreprise) et malgré tout interdépendantes.

• Maîtrise des accès personnalisés : Comment mettre à disposition les informations dont un utilisateur a besoin sans le noyer (l'immerger) sous des tonnes de données, sans le contaminer (infecter) avec des données dont il n'a que faire, et en garantissant le bon usage des informations. Ainsi, les étudiants peuvent avoir accès à leur résultat d'examen, mais ils ne peuvent pas les modifier.

• Maîtrise des traitements : Dès lors que la masse d'information est riche, complexe, en

constante évolution, il faut alors maîtriser le traitement de ces données. Ainsi l'élaboration de la paye, chez un commerçant employant quelques vendeurs et agents de service, sera sans comparaison avec celle d'une grande entreprise employant une grande variété de salariés sur quantité de régimes différents. Ces raisons vont nous amener vers la conception d'une base de données.

I.2.2. Le partage de l'information en fonction du type de système

Un SGBD doit offrir plusieurs interfaces d'accès, correspondant aux différents types d'utilisateurs pouvant s'adresser à lui. On trouve des interfaces orientées utilisateur comme le langage de requêtes déclaratifs : SQL (Structured Query Language), c'est un langage standard pour l'interrogation de bases de données.

Encore, une base de données est souvent indispensable dans le fonctionnement d'une organisation, et il n'est pas tolérable qu'une panne puisse remettre en cause son fonctionnement. Les SGBD fournissent des mécanismes pour assurer cette sécurité.

I.2.3. Architecture d'un SGBD

La plupart des SGBD suivent l'architecture ANSI/SPARC. Cette architecture est définie sur trois niveaux :

— Niveau externe
— Niveau conceptuel
— Niveau interne

Niveau externe : Retrouve toutes les possibilités d'accès par les différents utilisateurs. Correspond aux différents vues utilisateurs.

Niveau conceptuel : (logique) : permet de décrire la structure d'une BD globalement à tous les utilisateurs, il est produit par une analyse de l'application à modéliser. Le schéma conceptuel décrit la structure de la base indépendamment de son implantation.

Niveau interne : il s'appuie sur un système de gestion des fichiers pour définir la façon de stockage ainsi que le placement de données. Le SGBD doit être capable de faire des transformations entre chaque niveau, de manière à transformer une requête exprimée en terme du niveau externe en requête du niveau conceptuel puis du niveau physique.

I.2.4. Composants des SGBD

Un SGBD possède un certain nombre de composants logiciels et ce, quel que soit le modèle de données qu'il supporte. On trouve donc des composants chargés de :

La description des données : Cette partie sera constituée des outils (en gros des langages) permettant de décrire la vision des données de chaque utilisateur. On y trouve aussi les outils permettant de décrire le stockage physique des données.

La récupération des données : Cette partie prend en charge l'interrogation et la modification des données de façon optimisée. Elle est composée de langages de manipulation de données et gère aussi les problèmes de sécurité.

La sauvegarde et la récupération après pannes : Cette partie comporte des outils permettant de sauvegarder et de restaurer de façon explicite une base de données. Elle comporte aussi des mécanismes permettant, tant qu'une modification n'est pas finie, de pouvoir revenir à l'état de la base avant le début de cette modification.

Les accès aux données : C'est la partie chargée du contrôle de la concurrence des accès aux données. Elle doit être telle que chaque utilisateur attende le moins possible ses données tout en étant certain d'obtenir des données cohérentes en cas de mises à jour simultanées de la base.

Chapitre

II

Modèle Conceptuel : Modèle Entité/Association

II.1. Introduction

Le processus de conception d'une base de données BD suit les étapes suivantes :

— Analyse : c'est la perception du module réel par des discussions, des documentations
— Conception : proposer un schéma entité/association qui consiste à décrire le système ou le problème proposé
— Implémentation : c'est la transformation en modèle logique, hiérarchique, relationnel
— Utilisation : Concerne les étapes d'interrogation et de mise à jour
— Maintenance : Contient les étapes de correction et d'évolution

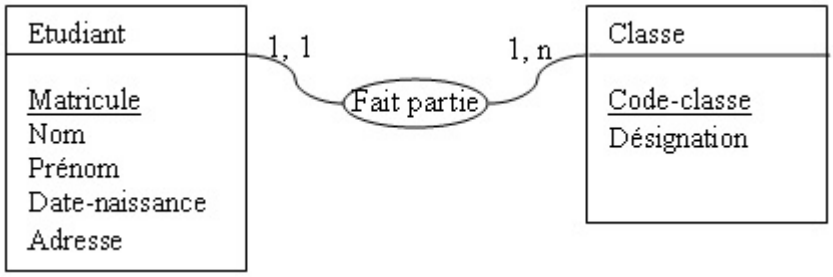
II.2. Modèle Entité/Association

a) Entité : une entité est un objet abstrait qui permet de regrouper des données de façon homogène ; ces données représentent le fonctionnement d'une organisation.

Exemple : l'organisation Faculté possède comme entité : étudiants, enseignants, classe ...

b) Association : elle représente le lien entre deux entités, cette relation correspond la plupart des cas à un verbe.

Le modèle Entité/Association utilise une représentation graphique. Cette représentation est donné par :

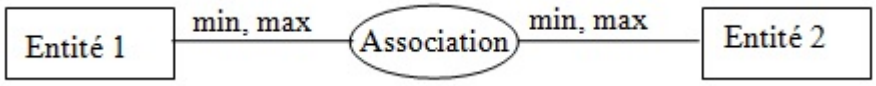


Ce schéma indique : « un étudiant fait partie d'une classe »

c) Propriétés : chaque entité est caractérisée par un ensemble de propriétés (attributs). L'entité étudiant à comme attributs : matricule, nom, prénom, date de naissance.

d) Identifiant : parmi les différentes propriétés, on a une qui sert à différencier sans ambiguïté chaque occurrence de l'entité. Cet identifiant doit être souligné dans la représentation graphique de l'entité.

e) Cardinalité : entre chaque entité et association, on indique deux cardinalité, une minimale et une maximale qui indiquent le nombre d'occurrence de l'entité qui peuvent être concerné par cette association.



Dans l'exemple précédent, nous pouvons lire les cardinalités de cette façon :

- Pour la cardinalité coté étudiant (1, 1) : Un étudiant appartient au minimum à une classe et au maximum à une classe.
- Pour la cardinalité coté classe (1, n) : Une classe possède au minimum 1 étudiant et au maximum n étudiant.

La notion de cardinalité minimum/maximum est liée aux types de liaison inter-entités.

- La cardinalité minimum est le nombre minimum d'occurrences d'une entité X
- La cardinalité maximum est le nombre maximum d'occurrences d'une entité X Les valeurs de cardinalités sont en général 0, 1 ou n :
- La cardinalité minimum à 0 veut dire que certaines occurrences de l'entité X ne sont pas impliquées dans une occurrence de l'association.
- La cardinalité minimum à 1 veut dire qu'une occurrence de l'entité X ne peut exister

sans participer à une occurrence de l'association.
- La cardinalité maximum à 1 veut dire que toute occurrence de l'entité X ne peut participer au plus qu'à une occurrence de l'association.
- La cardinalité maximum à n veut dire qu'une occurrence de l'entité X peut être impliquée dans un maximum de n occurrences de l'association.

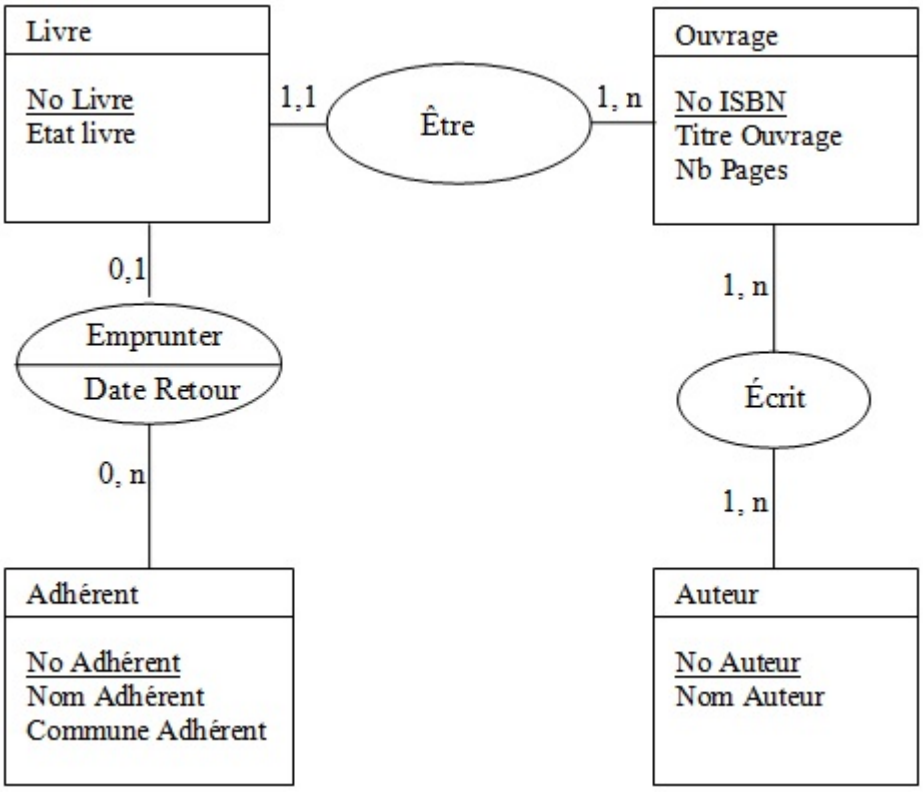
Les couples de cardinalité possible sont : (0, 1), (1, 1), (0, n) et (1, n).

II.3. Un exemple : Gestion d'une bibliothèque

Soit le dictionnaire des données suivant :
- numéro de livre : numéro attribué à chaque livre présent dans la bibliothèque
- numéro ISBN : numéro attribué à chaque ouvrage par une commission nationale. Un ouvrage peut exister en plusieurs exemplaires dans la bibliothèque. Chaque exemplaire ou livre a un numéro de livre interne à la bibliothèque
- état du livre : (3 valeurs : bon, moyen, mauvais)
- titre de l'ouvrage
- nombre de pages de l'ouvrage
- nom de ou des auteur(s)
- Numéro d'adhérent
- nom d'adhérent (en fait, nom et prénom dans une seule propriété)
- commune (municipalité, village) de l'adhérent
- chaque ouvrage est écrit par un ou plusieurs auteurs
- chaque auteur est caractérisé par Numéro auteur et nom auteur

Question : Déterminez le modèle conceptuel de données (modèle Entité/Association).

Réponse :
- La première étape est la détermination des entités et leurs propriétés (attributs).
- La deuxième étape est la mise en œuvre des associations et des cardinalités.


Livre
No Livre
Etat livre
1,1
Être
1, n
Ouvrage
No ISBN
Titre Ouvrage
Nb Pages
0,1
Emprunter
Date Retour
0, n
1, n
Écrit
1, n
Adhérent
No Adhérent
Nom Adhérent
Commune Adhérent
Auteur
No Auteur
Nom Auteur

Chapitre

III

Modèle Relationnel

III.1. Introduction

Le modèle relationnel dont les données sont stockées dans des tables a été formalisé par CODD en 1970. Le succès du modèle relationnel auprès des chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ses concepts. En outre, contrairement à certains autres modèles, il repose sur des bases théoriques solides, notamment la théorie des ensembles et la logique mathématique.

Les objectifs du modèle relationnel :

— proposer des schémas de données faciles à utiliser,

— mettre à la disposition des utilisateurs des langages de haut niveau pouvant éventuellement être utilisés par des non informaticiens,

— optimiser les accès à la base de données,

De façon informelle, on peut définir le modèle relationnel de la manière suivante :

— Les données sont organisées sous forme de tables, encore appelées relations et chaque

ligne appelé n-uplet ou tuple,
— Les données sont manipulées par des opérateurs de l'algèbre relationnelle,
— L'état cohérent de la base est défini par un ensemble de contraintes d'intégrités. Au modèle relationnel est associée la théorie de la normalisation des relations qui permet de se débarrasser des incohérences au moment de la conception d'une base de données.

III.2. Modèle Entité/Association

a) Relation Une relation est un sous-ensemble du produit cartésien de n domaines d'attributs (n > 0).

Une relation est représentée sous la forme d'un tableau à deux dimensions dans lequel les n attributs correspondent aux titres des n colonnes.

Une relation peut être représentée en intension ou en extension :
— Une représentation en intention est donnée par R (A1, A2, ..., An) Exemple : Etudiant (Matricule, Nom, Prénom, Date-naissance, Adresse, # Code-classe)
— Une représentation en extension donne une vision tabulaire (sous forme de table), dans lequel les n attributs correspondent aux titres des n colonnes

Exemple : Pour la relation Etudiant

Matricule	Nom	Prénom	Date-naissance	Adresse	Code-classe
12205230	Bali	Ali	20/01/1990	Tunis	TMEI3
12205500	Chtourou	Raja	25/09/1989	Sfax	TCI3

b) Schéma de relation Un schéma de relation précise le nom de la relation ainsi que la liste des attributs avec leurs domaines. Exemple : R (A1 :D1, A2 :D2, ..., An :Dn), avec Ai sont les attribut de la relations, et Di sont appelés domaines.

Un domaine est un ensemble de valeur atomique c'est-à-dire valeurs insécable (indivisible=non séparable). Pour simplifier l'écriture on peut omettre les domaines : R (A1, A2, ..., An).

c) Attribut Un attribut est un identificateur (un nom) décrivant une information stockée dans une base.

Exemples d'attribut : l'âge d'une personne, le nom d'une personne, le numéro de sécurité sociale.

d) Domaine Le domaine d'un attribut est l'ensemble, fini ou infini, de ses valeurs possibles.

Par exemple, l'attribut numéro de sécurité sociale a pour domaine l'ensemble des combinaisons de quinze chiffres et l'attribut nom a pour domaine l'ensemble des combinaisons

de lettres appelées chaîne de caractères.

e) Degré Le degré d'une relation est son nombre d'attributs.

f) Occurrence ou n-uplets ou tuples C'est un élément de l'ensemble figuré par une relation. Autrement dit, une occurrence est une ligne du tableau qui représente la relation.

g) Clé primaire La clé primaire d'une relation est l'identifiant de cette relation. Pour signaler la clé primaire, ses attributs sont généralement soulignés.

h) Clé étrangère Une clé étrangère dans une relation est formée d'un ou plusieurs attributs qui constituent une clé primaire dans une autre relation.

III.3. Passage du schéma E/A au schéma relationnel

Pour traduire un schéma du modèle Entité/Association vers le modèle relationnel, on peut appliquer les règles suivantes :

i. Chaque entité donne naissance à une relation. Chaque attribut de cette entité devient un attribut de la relation. L'identifiant est conservé en tant que clé de la relation.

ii. Chaque association, dont aucune liaison n'a pour cardinalité maximale 1 (de type n : n ; c'est-a-dire les deux cardinalités maximales sont à n), donne naissance à une relation. Chaque attribut de cette association devient un attribut de la relation. L'identifiant, s'il est précisé, est conservé en tant que clé de la relation, sinon cette clé est formée par la concaténation des identifiants des entités qui interviennent dans l'association.

iii. Chaque association binaire de type 1 : n ; c'est-a-dire elle possède une cardinalité maximale égale à 1 et l'autre cardinalité maximale égale n, ne devient pas une relation (généralement, cette association ne doit pas posséder d'attribut). Il décrit en effet une dépendance fonctionnelle (paragraphe dépendance fonctionnelle). La relation correspondant a l'entité dont la liaison vers l'association a une cardinalité maximale valant 1, se voit simplement ajouter comme attribut (et donc comme clé étrangère) l'identifiant de l'autre entité.

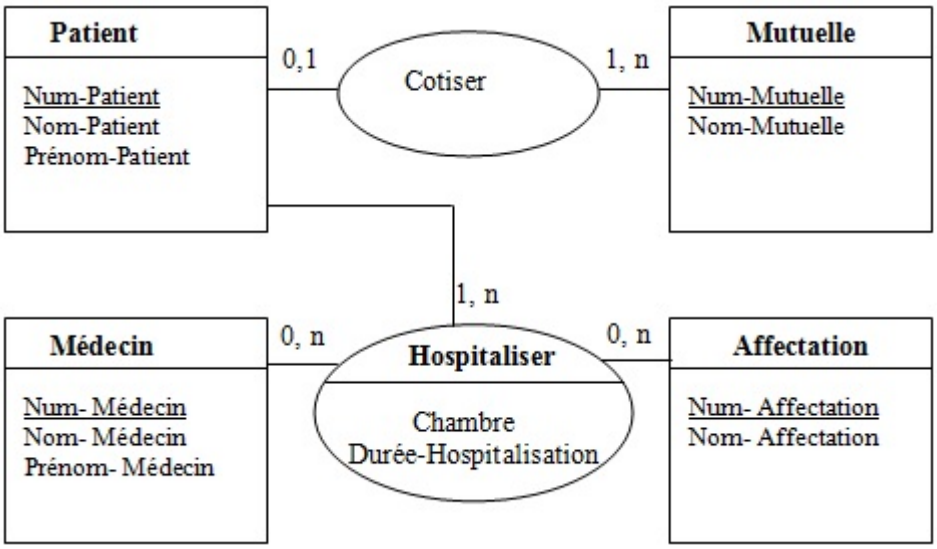
vi. Chaque association binaire de type 1 :1, c'est-a-dire dont les deux liaisons possèdent des cardinalités maximales à 1, ne devient pas une relation. Dans ce cas, la théorie impose qu'il doit y avoir au moins une coté de cardinalité (0,1). Si les deux cardinalités sont (0,1) et (1,1), alors la clé étrangère s'ajoute à l'entité du côté de la cardinalité (1,1). Par contre, si les deux cardinalité sont (0,1), alors la clé étrangère peut être placée indifféremment dans l'une des deux relations.

Remarque :

— une association binaire est une association entre deux entités

— une association binaire de type 1 : 1, si aucune des deux cardinalités maximales n'est n
— une association binaire de type 1 : n, si une des deux cardinalités maximales est 1 et l'autre est n

Exemple : Donner le modèle relationnel associé à ce modèle Entité/Association :

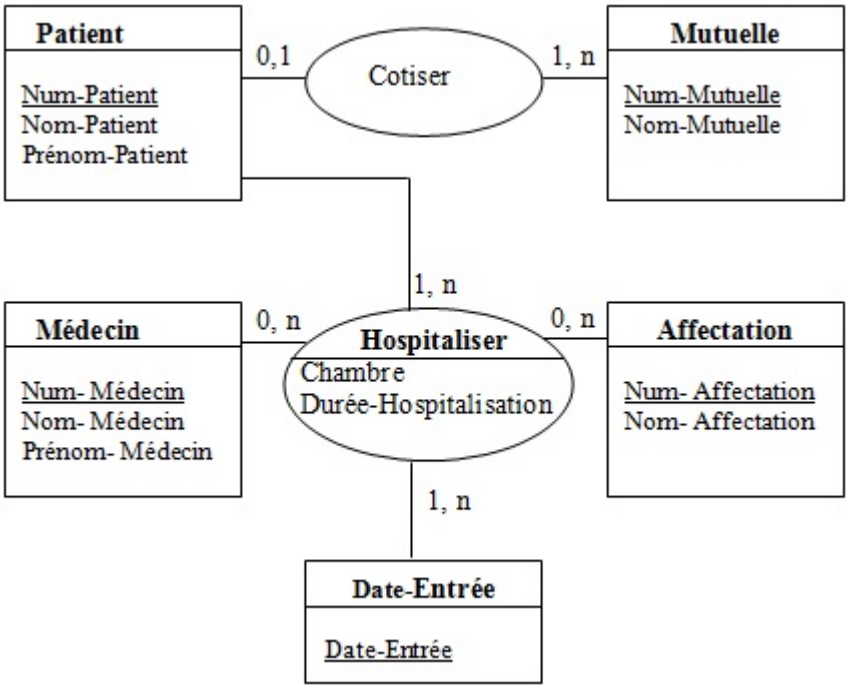


Le modèle relationnelle associé à cet exemple est :

— Patient (Num-Patient, Nom-Patient, #Num-Mutuelle)
— Mutuelle (Num-Mutuelle, Nom-Mutuelle)
— Médecin (Num-Médecin, Nom-Médecin, Prénom-Médecin)
— Affection (Num-Affection, Nom-Affection)
— Hospitaliser (#Num-Patient, #Num-Affection, #Num-Médecin, Chambre, Durée-Hospitalisation)

Cas Particuliers

Parmi les cas particuliers dans le passage du modèle Entité/Association vers le modèle relationnel, nous pouvons trouver le cas où une entité ne possède aucun attribut autre que sa clé. Lorsqu'une entité ne possède pas d'attribut en dehors de sa clé, il ne faut pas nécessairement en faire une relation. Comme l'entité *Date-Entré* dans l'exemple suivant :



Dans cet exemple, l'entité *Date-Entrée* ne doit pas se traduire en une relation. Le schéma relationnel adéquat correspondant au modèle Entité/Association est donc :

- Patient (Num-Patient, Nom-Patient, Num-Mutuelle)
- Mutuelle (Num-Mutuelle, Nom-Mutuelle)
- Médecin (Num-Médecin, Nom-Médecin, Prénom-Médecin)
- Affection (Num-Affection, Nom-Affection)
- Hospitaliser (#Num-Patient, #Num-Affection, #Num-Médecin, #Date-Entrée, Chambre, Durée-Hospitalisation)

III.4. Dépendance fonctionnelle

III.4.1. Définition

Une dépendance fonctionnelle (DF) est une contrainte qui doit vérifier des données de telle sorte que la base soit cohérente par rapport à la réalité qu'elle devrait représenter. On dit que les DF sont des contraintes d'intégrités de la BD.

Soit R (A1,A2,...An) un schéma de relation, et X et Y des sous ensembles de {A1,A2,...An}. On dit que X détermine Y ou Y dépend fonctionnellement de X si, et seulement si, des valeurs identiques de X impliquent des valeurs identiques de Y. On le note : $X \rightarrow Y$

Exemple : Numéro de la carte d'identité nationale (CIN) d'une personne nous renseigne sur son nom (Nom). On note donc : CIN→Nom

Autrement dit, il existe une dépendance fonctionnelle entre un ensemble d'attributs X et un ensemble d'attributs Y, que l'on note $X \rightarrow Y$, si connaissant une occurrence de X on ne peut lui associer qu'une seule occurrence de Y.

III.4.2. Propriétés des dépendances fonctionnelles

a) Réflexivité : Y$\subseteq$ X => X $\rightarrow$ Y

Tout ensemble d'attribut détermine lui même ou une partie de lui même.

b) Augmentation : X $\rightarrow$ Y => X, Z $\rightarrow$ Y, Z

X et Y peuvent être enrichis par un même 3ème ensemble (Z)

c) Transitivité : X $\rightarrow$ Y et Y $\rightarrow$ Z => X $\rightarrow$ Z

NV $\rightarrow$ TYPE et TYPE $\rightarrow$ PUISSANCE on déduit : NV $\rightarrow$ PUISSANCE

De ces 3 axiomes d'Armstrong on peut en déduire d'autres :

d) Union : X $\rightarrow$ Y et X $\rightarrow$ Z => X $\rightarrow$ Y, Z

e) Pseudo-transitivité : X $\rightarrow$ Y et W, Y $\rightarrow$ Z => W, X $\rightarrow$ Z

f) Décomposition : X $\rightarrow$ Y et Z $\subseteq$ Y => X $\rightarrow$ Z

Par la suite nous pouvons introduire les notions de dépendance fonctionnelle élémentaire et directe.

III.5. Formes normales

III.5.1. Définition

Normaliser une relation consiste à la représenter sous une forme canonique (universelle, standard). Les formes normales permettent d'éviter la redondance ou source d'anomalies dans les bases de données relationnelles afin d'éviter ou de limiter : les pertes de données, les incohérences au sein des données.

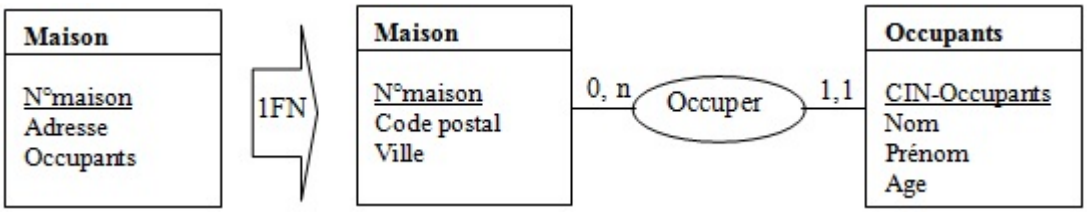
La normalisation conduit à la décomposition réversible de la relation non normalisée à un nombre d'autres relations. Ce qui consiste à isoler tout lien intra-relation. La réversibilité permet de retrouver la relation de départ, par des opérations de jointure. => Aucune perte d'information. On parle de décomposition sans perte. La décomposition doit être sans perte de DF, c-à-d que la fermeture transitive des DFs de R est la même que la fermeture transitive de l'union des Dfs des relations obtenues.

Plus le niveau de normalisation est élevé, plus le modèle est exempte de redondances. Une entité ou une association en forme normale de niveau n est automatiquement en forme normale de niveau n $-$ 1.

III.5.2. Première forme normale (1FN)

Une relation est en première forme normale (1FN) si tous ses attributs sont élémentaires, c'est-à-dire non décomposables.

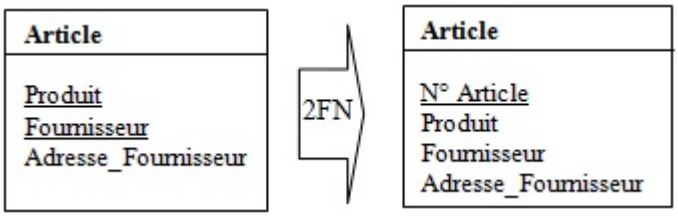
Un attribut composite doit être décomposé en attributs élémentaires (comme l'attribut Adresse dans l'exemple suivant) ou faire l'objet d'une entité supplémentaire (comme l'attribut Occupants).



L'élémentarité d'un attribut est toutefois fonction des choix de gestion. Par exemple, la propriété Adresse peut être considérée comme élémentaire si la gestion de ces adresses est globale. Par contre, s'il faut pouvoir considérer les codes postaux, les noms de rues, . . ., il convient d'éclater la propriété Adresse en Adresse (au sens numéro d'appartement, numéro et nom de rue, . . .), Code postal et Ville. En cas de doute, il est préférable (car plus général) d'éclater une propriété que d'effectuer un regroupement.

III.5.3. Deuxième forme normale (2FN)

Une relation est en deuxième forme normale (2FN), si et seulement si, elle est en première forme normale et si tout attribut n'appartenant pas à la clé dépend de la totalité de cette clé (tous les attributs non clés sont pleinement dépendants des clés). Autrement dit, les attributs doivent dépendre de l'ensemble des attributs participant à la clé. Ainsi, si la clé est réduite à un seul attribut, ou si elle contient tous les attributs, la relation est, par définition, forcément en deuxième forme normale. Ci-dessous un exemple de passage de la première forme normale vers la deuxième forme normale :



Cet exemple montre une entité Article décrivant des produits provenant de différents fournisseurs. On suppose qu'un même fournisseur peut fournir plusieurs produits et qu'un même produit peut être fourni par différents fournisseurs. Dans ce cas, les attributs Produit ou Fournisseur ne peuvent pas constituées un identifiant de l'entité Article. Par contre, le couple Produit/Fournisseur constitue bien un identifiant de l'entité Article. Cependant, l'attribut Adresse fournisseur ne dépend maintenant que d'une partie de la clé (Fournisseur). Opter pour une nouvelle clé arbitraire réduite à un seul attribut N°Article permet d'obtenir une entité Article en deuxième forme normale.

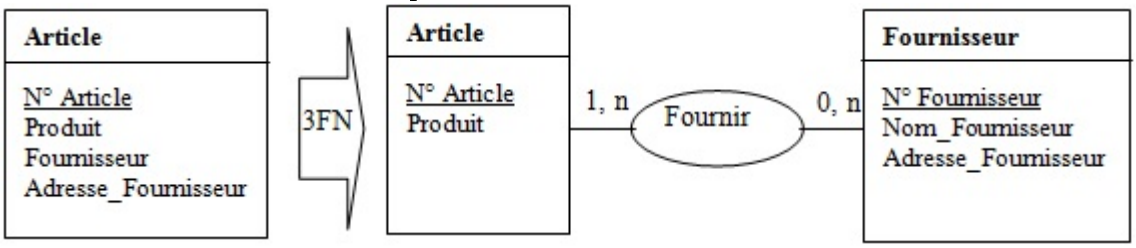
III.5.4. Troisième forme normale (3FN)

Une relation est en troisième forme normale (3FN), si et seulement si, elle est en deuxième forme normale et si tous ses attributs dépendent directement de sa clé et pas d'autres attributs (si tous les attributs non clés sont directement et pleinement dépendants

des clés).

Cette normalisation peut amener à désimbriquer (décomposé) des entités cachées comme le montre la figure suivante.

Une relation en deuxième forme normale avec au plus un attribut qui n'appartient pas à la clé est, par définition, forcément en troisième forme normale. Dans l'exemple suivant, l'attribut Adresse fournisseur dépend de l'attribut Fournisseur.



Comme solution donc, est de décomposer cette relation Article en deux relations : Article et Fournisseur. On donc pour la nouvelle relation Article, elle possède un seul attribut hors de sa clé, donc elle est en 3FN. Pour la relation fournisseur, elle possède deux attributs non clé, qui dépondent de la clé et sont indépendant entre eux, donc c'est une relation en 3FN. D'où, la solution finale est en 3FN.

III.6. Algorithme de décomposition

Entrée : un schéma relationnel (ensemble d'attributs) et un ensemble E de DF entre ses attributs

Sortie : une ou plusieurs relations en 3FN dont la jointure redonne la relation initiale (par contre des DF de E ont pu être perdues)

Principe : l'algorithme peut se voir comme la construction d'un arbre binaire. La racine de cet arbre est la relation à décomposer. L'arbre se construit récursivement de la manière suivante :

- on choisit une DF initiale « dfi » dans l'ensemble E des DF
- le fils gauche du nœud racine est une relation composé de tous les attributs de « dfi »
- dfi est retirée de l'ensemble E
- le fils droit du nœud racine est une relation composée de tous les attributs de la racine excepté ceux présents en partie droite de dfi

La solution dépend du choix des DF selon lesquelles on choisit de décomposer et il ne préserve pas nécessairement les DF. On sait néanmoins que toute relation admet une décomposition en 3FN qui préserve les DF.

Exemple :

Soit le schéma de relation R = {P, H, N, Y, T},

Soit E = {P $\rightarrow$ T ; P, H $\rightarrow$ Y ; H, N $\rightarrow$ P ; H, Y $\rightarrow$ N}, l'ensemble des dépendances fonctionnelles - L'ensemble des DF engendrées par E est le suivant :

H, N -> T P, H -> N H,N -> Y

H, Y -> P P, H -> T H, Y -> T

On a donc trois clés potentielles ((H, N) ; (P, H) ; (H, Y)) :

H, N -> P, T, Y

P, H -> T, Y, N

H, Y -> N, P, T

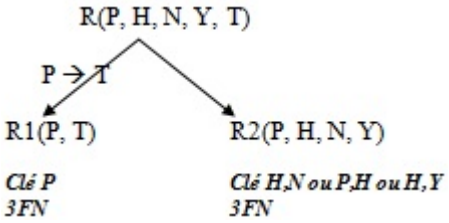
— les attributs clés sont donc : H, N, P, Y

— et les attributs non clés sont : T

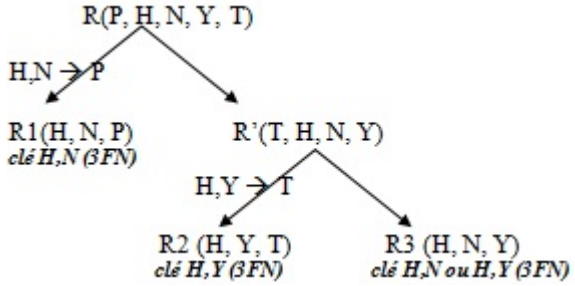
— par définition le schéma est en 1FN, mais pas en 2FN, car P, H -> T n'est pas une DF élémentaire (on à P -> T).

— R est donc en 1FN

— application de l'algorithme de décomposition :



Autrement :



Chapitre

IV

Langage de manipulation des données

IV.1. Introduction

Ces langages, dits assertionnels, sont basés sur la logique des prédicats d'ordre 1 et permettent de spécifier les données que l'on souhaite obtenir, sans dire comment y accéder. On doit y trouver des opérations permettant de :

- la recherche : retrouver des tuples (objets) vérifiant certains critères,
- l'insertion : ajouter des tuples,
- la suppression : enlever des tuples vérifiant certains critères,
- la modification : modifier des tuples vérifiant certains critères.

Un langage de manipulation de données n'est pas utilisable à lui seul, il doit aussi pouvoir être incorporable dans un langage de programmation classique.

On peut distinguer trois grandes classes de langages :

- Les langages algébriques basés sur l'algèbre relationnelle de CODD dans lesquels les requêtes sont exprimés comme l'application des opérateurs relationnels sur des

relations. C'est dans cette catégorie que l'on trouve le langage SQL (Structured Query Language), standard pour l'interrogation de bases de données.
- Les langages basés sur le calcul relationnel de tuples construits à partir de la logique des prédicats dans lesquels les variables manipulées sont des tuples.
- Les langages basés sur le calcul relationnel de domaines, construit aussi à partir de la logique des prédicats mais en faisant varier les variables sur les domaines des relations.

Le langage SQL comprend l'ensemble des instructions nécessaires à la spécification et à l'utilisation d'une base de données relationnelle. C'est un langage de type déclaratif c'est-à-dire que l'on spécifie les propriétés des données que l'on recherche et pas, comme dans un langage impératif, comment les retrouver.

Le langage SQL est à la fois :
- un langage d'interrogation de données (LID) : SELECT ;
- un langage de manipulation de données (LMD) : UPDATE, INSERT, DELETE ;
- un langage de définition des données (LDD) : ALTER, CREATE, DROP ;
- un langage de contrôle des données et des utilisateurs (LCD) : GRANT, REVOKE (casser).

IV.2. Langage d'interrogation de données (LID)

Le LID est un ensemble de commandes permettant la consultation des données :

Syntaxes :

SELECT [ALL / DISTINCT] attribut
FROM table(s)
WHERE condition
[GROUP BY attribut(s) [HAVING condition]]
[ORDER BY attribut(s) [ASC / DESC]]

- la clause SELECT permet de spécifier les attributs que l'on désire voir apparaître dans le résultat de la requête ; le caractère étoile (*) récupère tous les attributs de la table générée par la clause FROM de la requête ;
- la clause FROM spécifie les tables sur lesquelles porte la requête ;
- la clause WHERE, qui est facultative, énonce une condition que doivent respecter les n-uplets sélectionnés.
- la clause GROUP BY permet de définir des groupes
- la clause HAVING permet de spécifier un filtre (condition de regroupement des n-uplets) portant sur les résultats
- la clause ORDER BY permet de trier les n-uplets du résultat

Exemple : Donner la liste des tuples de tous les étudiants.

SELECT *

FROM etudiant ;

IV.2.1. Projection

L'opération de projection consiste à choisir le nom des colonnes de la (ou des) table(s) que l'on souhaite voir apparaître dans la réponse

SELECT attribut 1, attribut 2, . . . , attribut n

FROM table ;

Exemple : Donner le nom et prénom des étudiants avec leurs adresses.

SELECT nom, prenom, adresse

FROM etudiant ;

IV.3. Restriction ou Sélection

L'opération de sélection consiste à sélectionner des lignes (n-uplets, tuples) d'une (ou plusieurs) table(s) qui satisfont certaines conditions. Les conditions sont exprimées après la clause WHERE.

SELECT *

FROM table

WHERE condition ;

Exemple : Donner la liste des étudiants de la 3ème année.

SELECT *

FROM etudiant

WHERE Code-classe = '3ème année' ;

IV.3.1. Projection et Sélection

Les opérations de projection et de sélection peuvent évidemment être utilisées dans la même requête SQL

SELECT attribut 1, attribut 2, . . . , attribut n

FROM table

WHERE condition ;

Rq : on peut utiliser les opérateurs de comparaison =, >, !=, >=, <=, . . .

Exemples :

1) Donner le nom et prénom des étudiants âgé de moins de 23 ans.

SELECT nom, prenom

FROM Etudiant

WHERE ((Sysdate – date_naiss)/365,25)<23 ;

2) Donner les étudiants qui suivent une formation en section TMEI

SELECT Nom

FROM etudiant
WHERE Code-Classe = 'TMEI 1'
Or Code-Classe = 'TMEI 2'
Or Code-Classe = 'TMEI 3' ;

IV.3.2. Jointure

Une jointure est un produit cartésien entre deux relations par une condition. Il faut préciser, après la clause FROM, le nom des tables qui vont intervenir et, après la clause WHERE, les conditions qui vont permettre de réaliser la jointure.

Exemples : Produit cartésien entre Etudiant et Classe.

SELECT *
FROM Etudiant, Classe
WHERE Etudiant.Code-classe= Classe. Code-classe ;

Matricule	Nom	Code-classe	Code-classe	Désignation
80000000	Chtourou	TMEI1	TMEI1	Physique
80000001	Rekik	TMEI1	TMEI2	Physique
80000002	Ben Ali	TMEI3	TMEI3	Physique

IV.3.3. Les prédicats NULL, IN, LIKE, BETWEEN

a) Prédicat NULL Un attribut (une colonne, un champ...) peut avoir la valeur "NULL" soit en raison d'information incomplète (la valeur n'était pas connue au moment de la saisie des données) soit parce que la donnée n'est pas pertinente (juste). Dans une table qui modélise des individus, l'attribut "Nom Marital" n'est pas pertinent pour les individus de sexe masculin et a donc la valeur "NULL". La valeur "NULL" est différente de la valeur par défaut de l'attribut : zéro pour un attribut de type numérique et espace pour un attribut de type caractère.

La syntaxe est : IS NULL et sa négation IS NOT NULL.

Exemples : Donnez le numéro, nom et sexe des étudiants dont la date de naissance n'est pas connue

SELECT Matricule, Nom
FROM ETUDIANT
WHERE Date-naissance IS NULL ;

b) Prédicat IN Il comporte une liste de valeurs et vérifie si une valeur particulière apparaît sur cette liste.

La syntaxe est : IN (val1, val2, ...) et sa négation NOT IN (val1, val2, ...).

Lorsque la liste des valeurs est connue et fixe, le prédicat IN peut être remplacé par une suite d'opérateurs logiques OR.

c) Prédicat LIKE Ce prédicat permet de faire des recherches à l'intérieur d'une chaîne de caractères, lorsque l'on dispose d'informations incomplètes. Il utilise 2 caractères génériques :

— % : utilisé pour représenter une chaîne de caractère de longueur quelconque.

— _ : utilisé pour représenter un caractère unique.

La syntaxe est : LIKE 'Chaîne de recherche' et sa négation NOT LIKE 'Chaîne de recherche'

Exemples : Donnez le nom et date de naissance des étudiants dont la 2ème lettre du nom est 'a'.

```
SELECT Matricule, Date-naissance
FROM ETUDIANT
WHERE Nom LIKE '_a%';
```

d) Prédicat BETWEEN : Il permet de comparer la valeur d'un champ par rapport à une borne inférieure et une borne supérieure (bornes incluses).

Exemples : Donnez les noms des étudiants dont la matricule est comprise entre 6000 et 8000.

```
SELECT Nom
FROM Etudiant
WHERE Maricule BETWEEN 6000 AND 8000;
```

Ce prédicat, est une simplification d'écriture et peut être remplacé par :

```
SELECT Nom
FROM Etudiant
WHERE Maricule >= 6000 AND Maricule <= 8000;
```

IV.3.4. SQL : Regroupements

a) Clause GROUP BY Cette clause permet de créer des sous-ensembles (regroupements) de lignes pour lesquels la valeur d'une (ou plusieurs) colonne est identique. Cette clause est liée à l'utilisation de fonctions statistiques qui seront appliquées sur les sous-ensembles définis par GROUP BY.

Exemples : Regroupement par grade des enseignants

```
SELECT Grade
FROM ENSEIGNANT
GROUP BY Grade;
```

b) Clause HAVING Cette clause, contrairement à la clause WHERE qui précise les conditions à appliquer sur les lignes d'une table, permet de préciser des conditions au niveau des sous-ensembles créés par GROUP BY.

Exemples : Regroupement par grade des enseignants de type 'ASS'
SELECT Grade
FROM ENSEIGNANT
GROUP BY Grade HAVING Grade='ASS';

IV.4. Langage de définition des données (LDD)

Le langage de définition des données permet de définir les bases de données et les objets qui les composent : les tables, les vues, les index, les procédures.

La définition d'un objet peut être la création (CREATE), la modification (ALTER) et la suppression (DROP).

IV.4.1. Création

CREATE TABLE nom_table (nom_col1 TYPE1, nom_col2 TYPE2, ...), autrement
CREATE TABLE nom_table (Attribut1 TYPE1, Attribut2 TYPE2, ...)

Exemple : Donner la ligne de commande permettant de créer la table client possédant comme attributs Num-Client (Type=number(3), Clé primaire), Nom (Type=Char(30)), Date-Naissance (Type=Date), Salaire (Type=Number (8,2)), Num-Employé (Type=Number(3), Clé étrangère)

```
CREATE Table Client (Num-Client Number(3),
                     Nom Char (30),
                     Date-Naissance Date,
                     Salaire Number (8,2),
                     Num-Employé Number (3),
                     Constraint Clé_pri PRIMARY KEY (Num-Client)
                     Constraint Clé_etr FOREIGN KEY (Num-Employé));
```

IV.4.2. Modification

ALTER TABLE nom_table {ADD/MODIFY} ([nom_colonne type [contrainte], ...])
Exemples :

a) Renommer une colonne ALTER TABLE nom_table RENAME COLUMN ancien_nom TO nouveau_nom

b) Renommer une table ALTER TABLE nom_table RENAME TO nouveau_nom
c)

IV.4.3. Suppression

DROP TABLE nom_table

Exemples : DROP TABLE Client

Chapitre

V

Algèbre relationnelle

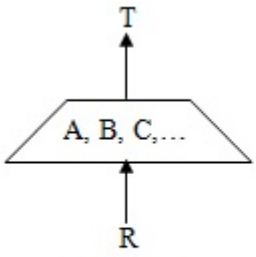
V.1. Introduction

L'algèbre relationnelle est un support mathématique cohérent sur lequel repose le modèle relationnel. L'objet de cette section est d'aborder l'algèbre relationnelle dans le but de décrire les opérations qu'il est possible d'appliquer sur des relations pour produire de nouvelles relations. L'approche suivie est donc plus opérationnelle que mathématique.

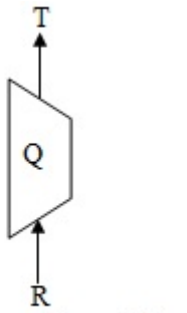
On peut classifier les opérateurs relationnels en trois catégories : Avec R et S sont les deux relations d'entrées, et T représente la relation de sortie.

V.1.1. Opérateurs unaires (sélection, projection)

se sont les opérateurs les plus simples, ils permettent de produire une nouvelle table à partir d'une autre table.



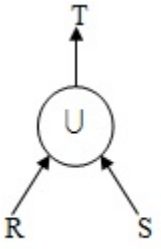
Projection



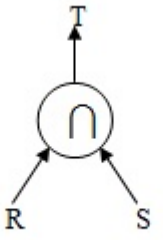
Sélection ou Restriction

V.1.2. Opérateurs binaires travaillant sur des relations de même schéma (union, intersection, différence)

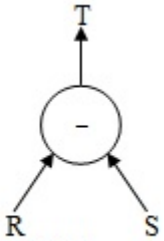
ces opérateurs permettent de produire une nouvelle relation à partir de deux relations de même degré et de même domaine.



Union



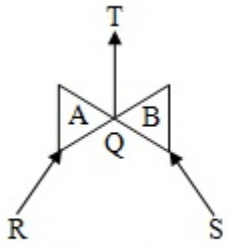
Intersection



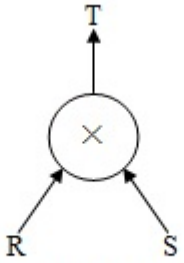
Différence

V.1.3. Opérateurs binaires travaillant sur des relations de schémas différents (jointure, produit cartésien, division)

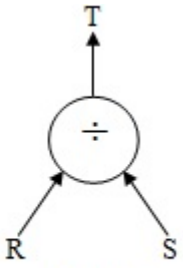
Ces opérateurs permettent de produire une nouvelle table à partir de deux ou plusieurs autres tables.



Jointure



Produit Cartésien



Division

V.2. Opérateurs de l'algèbre relationnelle

V.2.1. Opérateurs unaires

— **Projection :** C'est le découpage "vertical" d'une relation

La projection consiste à supprimer les attributs autres que A1, ..., An d'une relation et à éliminer les n-uplets en double apparaissant dans la nouvelle relation.

En d'autres termes, la projection permet de choisir des colonnes dans le tableau. Si R est vide, la relation qui résulte de la projection est vide, mais pas forcément équivalente (elle contient généralement moins d'attributs).

Le tableau suivant montre un exemple de la relation Étudiant :

Matricule	**Nom**	**Prénom**
5000	Dhouib	Fatma
6000	Trabelsi	Majdi
7000	Zouch	Wassim
8000	Trabelsi	Wassim

Un exemple de Projection sur la colonne nom de la relation Étudiant, est donné par le tableau suivant : (liste des noms de la table Étudiant)

Nom
Dhouib
Trabelsi
Zouch

Les tuples redondants sont supprimés

— **La Sélection ou Restriction :** Découpage "horizontal" d'une relation

La sélection (parfois appelée restriction) génère une relation regroupant exclusivement toutes les occurrences de la relation R qui satisfont l'expression logique Q.

En d'autres termes, la sélection permet de choisir des lignes dans le tableau. Le résultat de la sélection est donc une nouvelle relation qui a les mêmes attributs que R. Si R est vide (i.e. ne contient aucune occurrence), la relation qui résulte de la sélection est vide.

Exemples : Quels sont les Étudiants dont la matricule est 5000 ou 7000 ?

Matricule	**Nom**	**Prénom**
5000	Dhouib	Fatma
7000	Zouch	Wassim

V.2.2. Opérateurs binaires de même schéma

Les trois opérateurs ensemblistes opèrent sur des relations R et S de même schéma

— **Union :** L'union est une opération portant sur deux relations R et S ayant le même schéma et construisant une troisième relation constituée des n-uplets appartenant à chacune des deux relations R et S sans doublon, on la note : T = R$\cup$S

Comme nous l'avons déjà dit, R et S doivent avoir les mêmes attributs et si une même occurrence existe dans R et S, elle n'apparaît qu'une seule fois dans le résultat de l'union. Le résultat de l'union est une nouvelle relation qui a les mêmes attributs que R et S. Si R et S sont vides, la relation qui résulte de l'union est vide. Si R (respectivement S) est vide, la relation qui résulte de l'union est identique à S (respectivement R).

Exemple :

Relation R	
Nom	**Prénom**
Dhouib	Fatma
Trabelsi	Majdi
Zouch	Wassim
Trabelsi	Wassim

Relation S	
Nom	**Prénom**
Trabelsi	Wassim
Feki	Imed
BenHamida	Ahmed

Relation T	
Nom	**Prénom**
Dhouib	Fatma
Trabelsi	Majdi
Zouch	Wassim
Trabelsi	Wassim
Feki	Imed
BenHamida	Ahmed

— **Intersection :** L'intersection est une opération portant sur deux relations R et S ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux appartenant aux deux relations, on note : T = R ∩ S Comme nous l'avons déjà dit, R et S doivent avoir les mêmes attributs. Le résultat de l'intersection est une nouvelle relation qui possède les mêmes attributs que R et S. Si R ou S ou les deux sont vides, la relation qui résulte de l'intersection est vide.

Exemple :

Relation R	
Nom	**Prénom**
Dhouib	Fatma
Trabelsi	Majdi
Zouch	Wassim
Trabelsi	Wassim

Relation S	
Nom	**Prénom**
Dhouib	Fatma
Feki	Imed
Trabelsi	Wassim

Relation T	
Nom	**Prénom**
Dhouib	Fatma
Trabelsi	Wassim

— **Différence :** La différence est une opération portant sur deux relations R et S ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux ne se trouvant que dans la relation R. On note R – S.

Comme nous l'avons déjà dit, R et S doivent avoir les mêmes attributs. Le résultat de la différence est une nouvelle relation qui a les mêmes attributs que R et S. Si R est vide, la relation qui résulte de la différence est vide. Si S est vide, la relation qui résulte de la différence est identique à R.

Exemple :

Relation R	
Nom	**Prénom**
Dhouib	Fatma
Trabelsi	Majdi
Zouch	Wassim
Trabelsi	Wassim

Relation S	
Nom	**Prénom**
Dhouib	Fatma
Feki	Imed
Trabelsi	Wassim

Relation T	
Nom	**Prénom**
Trabelsi	Majdi
Zouch	Wassim

V.2.3. Opérateurs binaires de schémas différents

Soient R et S deux relations définies sur les schémas SR et SS.

— **Produit cartésien (R X S) :** Le produit cartésien est une opération portant sur deux relations R et S et construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R et S, on note R x S.

Le résultat du produit cartésien est une nouvelle relation qui a tous les attributs de R et tous ceux de S. Si R ou S ou les deux sont vides, la relation qui résulte du produit cartésien est vide. Le nombre d'occurrences de la relation qui résulte du produit cartésien est le nombre d'occurrences de R multiplié par le nombre d'occurrences de S.

Relation R

Nom	Prénom
Dhouib	Fatma
Trabelsi	Majdi

Relation S

Article	Prix
Livre	45
Montre	25
Téléphone	100

Relation T

Nom	Prénom	Article	Prix
Dhouib	Fatma	Livre	45
Dhouib	Fatma	Montre	25
Dhouib	Fatma	Téléphone	100
Trabelsi	Majdi	Livre	45
Trabelsi	Majdi	Montre	25
Trabelsi	Majdi	Téléphone	100

— **Division :** La division est une opération portant sur deux relations R et S, telles que le schéma de S est strictement inclus dans celui de R, qui génère une troisième relation regroupant toutes les parties d'occurrences de la relation R qui sont associées à toutes les occurrences de la relation S ; on note : T = R÷S

Autrement dit, la division de R par S (R ÷ S) génère une relation qui regroupe tous les n-uplets qui, concaténés à chacun des n-uplets de S, donne toujours un n-uplet de R. La relation S ne peut pas être vide. Tous les attributs de S doivent être présents dans R et R doit posséder au moins un attribut de plus que S (inclusion stricte). Le résultat de la division est une nouvelle relation qui a tous les attributs de R sans aucun de ceux de S. Si R est vide, la relation qui résulte de la division est vide.

Relation R

Enseignant	Etudiant
Mohamed	Fatma
Mourad	Majdi
Salem	Ali
Kacen	Ammar

Relation S

Etudiant
Fatma
Majdi

Relation T

Enseignant
Mohamed
Mo

— **Jointure :** La jointure est une opération portant sur deux relations R et S qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences de relations R et S qui satisfont l'expression logique La jointure est notée : T = R ►◄ S. La jointure est équivalente à un produit cartésien

suivi d'une opération de sélection (qualification).

Bibliographie

[1] C. Gruau, "Conception d'une base de données," 2006.

[2] I. Akoka, J. & Comyn-Wattiau, "Conception des bases de données relationnelles. vuibert informatique," 2001.

[3] J. Gabillaud, "Sql et algèbre relationnelle - notions de base," *ENI*, 2004.

[4] R. Godin, "Systèmes de gestion de bases de données," *Loze-Dion*, vol. 1&2, 2000b.

[5] D. Marre, "Introduction aux systèmes de gestion de bases de données," 1996.

[6] L. Petrucci, "Base de données. présentation projetée et travaux dirigés," 2006.

[7] J.-M. Saglio, "Dominante informatique : Module bases de données." 2002.