الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
Ecole Supérieure en Génie
Electrique et Energétique d'Oran
Département du cycle préparatoire
ESG2E ORAN
المدرسة العليا في الهندسة الكهربائية و الطاقوية
بوهران
قسم التكوين التحضيري

POLYCOPIÉ
COURS ET EXERCICES

Informatique 4

INTRODUCTION AUX BASES DE DONNÉES

Selon le programme pédagogique des classes préparatoires en sciences et technologies (juillet 2015)

Rédigé par
Dr. TANDJAOUI Amel Faiza

Octobre 2022

TABLE DES MATIÈRES

CHAPITRE 0

INTRODUCTION GÉNÉRALE

0.1 Introduction

Le module "informatique 4" des classes préparatoires en sciences et technologies est une initiation au concept de base de données.

L'information représente de nos jours une forme de puissance. Que ce soit dans le milieu professionnel ou personnel, elle est de plus en plus accessible à travers l'accès quasi immédiat à bon nombre de bases de données, notamment grâce à Internet.

Ce module a pour but d'initier le futur ingénieur aux concepts de mise en place d'une base de données, en partant du principe de conception, jusqu'à la prise en main du fameux langage SQL pour la gestion de base de données.

Nous nous baserons pour ce cours sur la méthode Merise pour la conception et le développement de systèmes d'informations informatisés [1] dont feront parti les bases de données étudiées. Cette méthode a été établie en 1978 par une équipe de chercheurs français et reste jusqu'à ce jour très répondue. La méthode Merise repose sur la décomposition du processus de création d'un système d'information en plusieurs niveaux d'abstraction, c'est à dire des niveaux où on prendra en considération certains détails du système mais pas d'autres, mettant à disposition du concepteur plusieurs outils de modélisation. La méthode Merise repose également sur le principe de séparation de la modélisation entre les données et les traitements. Notre cours portera uniquement sur la partie données.

Parmi les modèles de données, i.e. forme de stockage de données, existants, nous nous intéresserons au modèle relationnel où tout est stocké sous forme de tableaux à deux dimensions, dits tables ou bien relations. Une algèbre spécialement conçue pour l'application d'opérations sur ces relations et en obtenir de nouvelles a été développée. Il s'agit de l'algèbre relationnelle.

Ce cours est structuré comme suit : l'actuel chapitre introduit les concepts clés liés à la notion de base de données. Le chapitre 1 présente deux modèles de la méthode Merise, à savoir le modèle conceptuel de données et le modèle logique de données. Le chapitre 2 traite quant à

lui les principaux opérateurs de l'algèbre relationnelle. Enfin, le chapitre 3 donne un aperçu du langage SQL.

0.2 Définitions

0.2.1 Système d'information

Un système d'information est un ensemble organisé de ressources qui permet de collecter, stocker, traiter et distribuer de l'information. Ces ressources sont : du personnel, du matériel, des logiciels, des données et des procédures. Le système d'information d'une organisation doit s'adapter à sa stratégie.

0.2.2 Base de données

Les bases de données sont l'un des constituants d'un système d'information. Elles peuvent être définies comme suit :

Une base de données (BDD) est une collection de données ***reliées****, stockées sur* ***ordinateur****, et qui peuvent être utilisées par* ***plusieurs applications*** *sans avoir à connaître leurs* ***détails de stockage*** [2].

Les BDDs peuvent être de diverses natures. Une base de données très simple peut être un carnet d'adresses d'une personne : ensemble d'informations sur ses contacts (nom, prénom, adresse, numéros de téléphone, adresse e-mail, etc.). Une base de données plus complexe serait au niveau d'une usine de fabrication de véhicules où on aurait besoin de gérer la liste de toutes les pièces nécessaires, de savoir pour quels modèles de véhicules elles sont utilisées, quelle est leur quantité en stock, etc.

0.2.3 Système de gestion de bases de données

Un système de gestion de base de données (SGBD) est un système logiciel qui gère une base de données, en permettant, notamment, d'effectuer dessus les opérations telles que le stockage, la consultation, la recherche, l'insertion, la mise à jour, etc. [3].

Parfois les données recherchées par l'utilisateur d'une BDD ne sont pas celles qui sont directement stockées dans cette BDD. Ainsi, le SGBD doit fournir le moyen d'appliquer des formules mathématiques par exemple afin de pouvoir déduire des informations à partir des données stockées. Ces formules peuvent être simples : calcul de la moyenne d'une série de données, ou plus complexe : interpolation/extrapolation pour la déduction d'estimations de données inexistantes dans la BDD.

Il existe plusieurs types de SGBDs. Dans notre cours, nous nous intéressons aux SGBDs de type relationnel (SGBDR) -voir la section "Modèles de données"-.

0.3 Le cycle de vie d'une BDD

Le cycle de vie d'une BDD représente l'ensemble des étapes par lesquelles celle-ci passera. Comme le montre le schéma simplifié donné à la figure 1, il comprend les étapes suivantes :

1. Analyse des besoins : étape où il faudra récolter et étudier tout ce qu'attend l'utilisateur du système de base de données.
2. Conception : étape où la BDD est modélisée selon les besoins analysés lors de l'étape précédente. Un *modèle conceptuel* est tout d'abord proposé. Ce sera un diagramme qui reprendra les "objets" qui seront stockés dans la base de données et leurs relations. Ce modèle devra être normalisé, c.à.d. qu'il devra respecter un certain nombre de règles afin de pouvoir produire une BDD efficace. Un *modèle logique* décrira par la suite l'organisation des données dans la BDD. Puis, un *modèle physique* viendra décrire les détails de stockage des données sur le support physique, c.à.d. sur le matériel. Dans ce cours, nous nous intéresserons seulement aux deux premiers modèles, c'est-à-dire, le modèle conceptuel et le modèle logique. Ce sera l'objet du chapitre 2.
3. Implémentation : étape où la base de données est programmée et donc créée en utilisant un langage de programmation tel que le langage SQL à travers un SGBD. Une introduction au langage SQL se fera au chapitre 3.
4. Utilisation et maintenance : étape qui comprend l'insertion des données dans la BDD, leur modification et leur suppression. Une fois les données insérées, des recherches peuvent être effectuées dessus par l'utilisateur. Selon le système considéré, tout ceci pourra se faire à travers des requêtes écrites dans un langage de programmation tel que le SQL, ou bien à travers une interface. Cette étape comprend également une maintenance du système par un administrateur avec des opérations telles que la création de comptes sécurisés pour les utilisateurs, la mise en place de sauvegardes du contenue du système, etc.

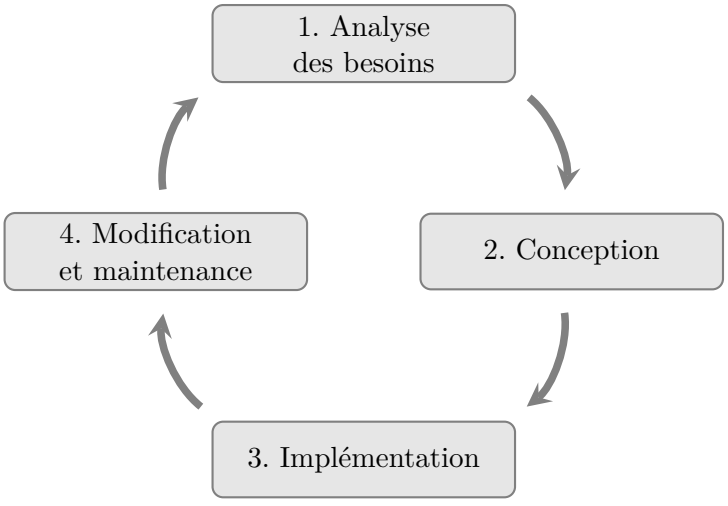


Figure 1 – Cycle de vie simplifié d'une base de données.

Remarque : Le diagramme de la vie d'une BDD est représenté par un cycle, car au bout d'un certain temps d'utilisation, la BDD aura besoin d'évoluer. Les étapes précédemment décrites se répèteront alors.

0.4 Rôle dans une BDD : Qui fait quoi ?

Dans la vie d'une BDD, plusieurs personnes, avec des rôles différents, pourront intervenir. Nous citerons les 4 rôles principaux suivants :

0.4.1 L'utilisateur

C'est la personne qui va utiliser les données de la base de données. Il peut y avoir plusieurs types d'utilisateurs selon le besoin de chacun. Il est possible qu'un certain utilisateur n'ait le droit que de faire des recherches sur la BDD, comme il est possible qu'il y ait un utilisateur qui puisse effectuer des insertions, modifications et recherches sur certaines données mais pas sur d'autres, ou encore utilisateur qui a le droit d'effectuer toutes les opérations possibles sur toutes les données de la BDD.

0.4.2 L'administrateur

L'administrateur de la base de données est la personne qui est en charge de la maintenance, des opérations, de la sécurité, de la confidentialité et de l'efficacité de la base de données.

L'administrateur peut également aider les utilisateurs à apporter des modifications/évolution à la base de données au besoin.

0.4.3 Le concepteur

Le concepteur est en charge de concevoir la structure générale de la BDD et de son interface, suite à la collecte et à l'étude des besoins des utilisateurs et de la nature des données à stocker. C'est à lui de définir la structure de la BDD.

0.4.4 Le développeur

Le développeur est celui qui programme la BDD avec son interface, à travers un ou plusieurs langages de programmation.

Remarque

Selon la nature de la BDD, il est possible que les rôles cités ci-dessus se chevauchent, et/ou qu'il y ait plus de rôles. Pour une petite BDD par exemple, il est possible qu'une personne soit en même temps : conceptrice, développeuse, et seule utilisatrice de la BDD. Comme il se peut aussi que l'administration de la BDD soit gérée par toute une équipe dans le cas de grandes sociétés.

0.5 Architecture d'une base de données

L'architecture d'une BDD représente l'ensemble des schémas qui spécifient différents niveaux de vue de la base de données. On peut considérer que l'architecture d'une BDD comporte 4 niveaux, allant du niveau le plus bas, et donc le plus proche du matériel qui est la niveau physique, au niveau le plus haut qui est le plus proche de l'utilisateur, le niveau externe, comme le montre la figure 2.

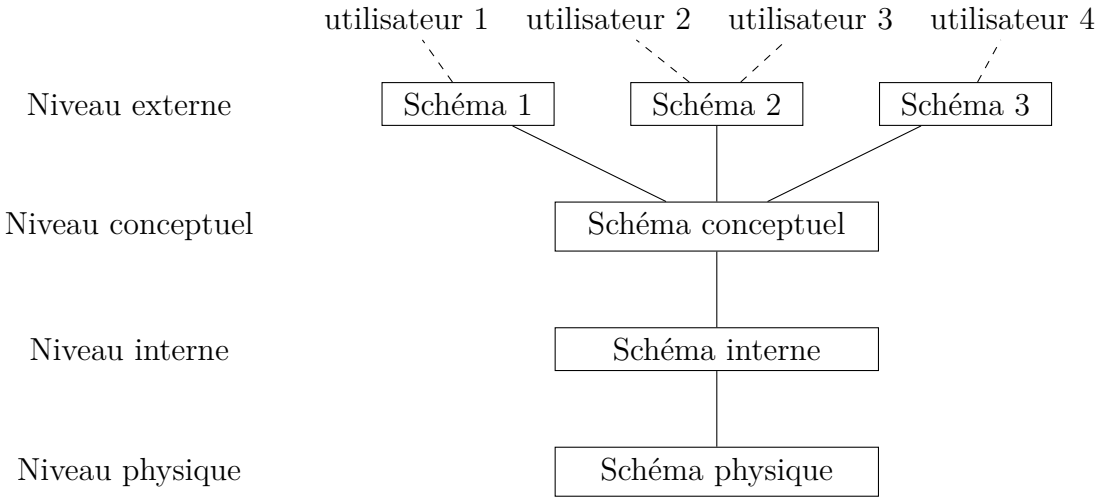


FIGURE 2 – Architecture d'une base de données [3].

0.5.1 Schéma physique

Niveau le plus bas, c'est-à-dire, le plus proche du matériel. Il décrit les données tels que stockées sur le support physique utilisé et permet de savoir exactement leur emplacement sur ce dernier. L'utilisateur d'une base de données n'a généralement pas à se préoccuper de ce niveau.

0.5.2 Schéma interne

Ce niveau voit les données en termes de fichiers logiques et d'enregistrements et en termes de mécanisme d'accès utilisé pour trouver les enregistrements [3]. Un fichier est donc vu comme un ensemble d'enregistrements numérotés, en faisant abstraction de la manière avec laquelle ces données sont stockées physiquement sur le disque. C'est au SGBD de faire le mapping entre le schéma physique et le schéma interne.

0.5.3 Schéma conceptuel

C'est une vue globale de toutes les données dans la BDD. Il décrit à travers un modèle de données un réel qui peut représenter les informations liées à une entreprise, à une partie d'une entreprise, etc. Dans le domaine des sciences et de l'ingénierie, il peut décrire des informations

liées à une partie du monde réel, ex : propriétés mécaniques d'alliages. Le schéma conceptuel est généralement réalisé par l'utilisateur.

0.5.4 Schéma externe

Celui-ci représente la vue d'un utilisateur à partir du schéma conceptuel. Il en est donc une partie et s'adapte aux besoins de l'utilisateur, selon les types de requêtes/questions et de données qu'il devra manipuler. Un seul schéma conceptuel pourra donc avoir des schémas externes différents afin de s'adapter aux besoins de plusieurs classes d'utilisateurs. Ainsi, un schéma conceptuel de données pourrait décrire les propriétés de matériaux à travers une table avec les colonnes suivantes : Matériau, température, résistivité électrique, permittivité magnétique, conductivité thermique, permittivité, capacité thermique. Dans la table on pourrait avoir plusieurs matériaux, et pour chaque matériau, des mesures pour plusieurs températures. En tant qu'ingénieur en génie électrique, on serait intéressé par les propriétés électriques et magnétiques des alliages de métaux à température ambiante. La vue de la base de données consistera alors en les colonnes suivantes seulement : Alliage, température, résistivité électrique, permittivité magnétique. C'est tout ce qu'on verra de la BDD. Les autres données seraient comme inexistantes. Et toutes les requêtes (questions/modification) faites devront se faire uniquement sur ces données là. Il est alors impossible de modifier des données dont on n'a pas l'accès, par erreur. Dans le cas où tous les utilisateurs auraient besoin de voir tout le contenu de la base de données, alors le schéma externe n'est plus nécessaire. Cela est généralement le cas pour les petites bases de données.

0.5.5 Interface utilisateur

Une fois la vue utilisateur choisie, le SGBD permet l'accès à la base de données soit à travers un langage de requête spécial -SQL étant le plus répondu-, ou à travers un code intégré dans le programme d'utilisateur. L'affichage des résultats pour l'utilisateur peut être plus ou moins élaboré afin d'être simple à déchiffrer (tableau, forme graphique, textes, etc). Ainsi, l'interface répond aux besoins de l'utilisateur c'est pourquoi il doit être apte à les exprimer clairement.

0.6 Modèle de données

Un modèle de données pour un système de base de données détermine la structure générale du SGBD. Une fois identifié, toutes les données de la BDD seront organisées selon la même structure.

Les 3 principaux modèles sont :

— Le modèle relationnel où les données sont représentées par des tables ;

— Le modèle hiérarchique où les données sont représentées par des arbres ;

— Le modèle réseau où les données sont représentées par un ensemble d'arbres à deux niveaux.

Dans ce module, nous nous intéresserons au modèle relationnel.

0.6.1 Modèle relationnel

Une base de données basée sur le modèle relationnel est une *base de données relationnelle*. Les origines du modèle relationnel remontent à l'année 1969, mais il reste jusqu'à aujourd'hui le modèle de base sur lequel repose la majorité des bases de données. Le langage SQL prend ses bases dans le modèle relationnel.

Dans une base de données relationnelle, les données sont représentées sous forme de tableaux à deux dimensions. Chaque tableau est appelée *relation* ou *table*. Une base de données relationnelle est donc un ensemble de relations.

Chaque ligne dans une relation est appelée un *n-uplet* ou *tuple* et le titre de chaque colonne est appelé *attribut*. Une donnée à l'intersection d'une ligne et d'une colonne est appelée *composant* du tuple. Un *domaine* d'un attribut est la plage de valeurs qui peuvent être prises par les composants appropriés. Une relation R avec une liste d'attributs $A_1, A_2, \ldots, A_m$ sera décrite par son *schéma* : $R(A_1, A_2, \ldots, A_m)$. Le *schéma d'une base* de données est l'ensemble des schémas des tables de cette base de données.

Prenons la relation "Atome_masse" suivante :

Atome	Symbole	Numéro_atomique	Masse_atomique
Hydrogène	H	1	1,008
Helium	He	2	4,003
Beryllium	Be	3	9,015

La liste des attributs de cette relation est : Atome, Symbole, Numéro_atomique, Masse_atomique. Le premier tuple est constitué des composants suivants : Hydrogène, H, 1 et 1,008. Le domaine de l'attribut Numéro_atomique est l'ensemble $\{1, 2, 3\}$. Le schéma de la relation est Atome_masse(Atome, Symbole, Numéro_atomique, Masse_atomique).

0.7 Système de gestion de base de données

Le système de gestion de base de données (SGBD) est le logiciel qui contrôle toute la base de données. Il est responsable des opérations et des mapping (transformations nécessaires pour passer d'un schéma à l'autre). Le SGBD peut faire appel au système d'exploitation au-dessus duquel il est installé pour certaines opérations (le mapping entre schéma interne et le schéma physique par exemple). L'utilisateur n'aura pas à se préoccuper de cela.

0.7.1 Langage de données

Un SGBD propose des langages permettant d'interagir avec les données. Ces langages peuvent être classés comme suit :

Le langage de requête

Le langage de requête est généralement le seul langage que l'utilisateur voit. Il peut être de différentes formes selon l'interface proposée à l'utilisateur, ex. des questions exprimées en langage naturel, des séries de sélections à choix multiples, etc.

Un exemple de requête simple exprimée en langage naturel : Quelle est la masse atomique du plomb selon la base de données ?

Le langage de manipulation de données (LMD)

Le langage de manipulation de données exprime les requêtes de l'utilisateur sous forme de commandes. La requête exprimée en langage naturel dans la section précédente peut être traduite dans le langage SQL comme suit afin de pouvoir être exécutée sur la BDD :

```
SELECT masse FROM masse_atome WHERE nom_atome="plomb";
```

qui signifie qu'on voudrait sélectionner la valeur de la colonne masse atomique à partir de la table qui s'appelle masse_atome dans la ligne où le nom de l'atome est égal à plomb.

Le langage de manipulation de données permet également d'insérer des données, de les modifier et de les supprimer de la base de données.

Le langage de définition de données (LDD)

Avant de pouvoir manipuler des données, il est nécessaire de décrire la forme qu'elles vont avoir. Ainsi, le SGBD devra notamment connaitre, la liste des attributs/colonnes de chaque table et leur type, ainsi que les contraintes sur les données.

Un exemple de contrainte est le fait que le symbole d'un atome dans la table masse_atome doit toujours comporter un premier caractère majuscule qui peut être suivi par un seul autre caractère en minuscule.

Le LDD permet également de créer des vues d'utilisateurs.

Le langage de contrôle de données (LCD)

Ce langage contrôle l'accès aux données stockées dans la base en permettant de spécifier les droits et les permissions des différents utilisateurs.

Le langage de contrôle de transactions (LCT)

Ce langage gère les transactions d'une base de données. Les transactions étant un ensemble de requêtes reliées (ex : ensemble de requêtes nécessaires pour effectuer un virement bancaire d'un compte vers un autre compte).

0.7.2 Sécurité, confidentialité et statistiques

Un SGBD doit garantir la sécurité des données notamment en réalisant des sauvegardes de la base de données qu'un utilisateur peut recharger afin de revenir à une version antérieure de la base de données et en s'assurant qu'un utilisateur n'obtienne pas de données inconsistantes car un autre utilisateur serait en train de modifier les mêmes données.

Un SGBD peut garantir la confidentialité des données afin que chaque utilisateur n'accède qu'aux données auxquelles il a le droit d'accéder. À titre d'exemple, un musée peut rendre une partie de sa base de données accessible au public, mais ne souhaite pas que la valeur des objets soit consultable afin qu'on ne sache pas quels objets sont précieux.

Un SGBD peut également fournir des statistiques sur l'utilisation des données. Cela permet notamment à l'administrateur de la base de données d'optimiser le système afin de rendre rapides les requêtes les plus courantes et les plus importantes.

0.7.3 Dictionnaire de données

Un SGBD donne accès à un dictionnaire de données qui fournit des informations sur les données de la BDD : nom des fichiers, noms des attributs, leurs formats, les contraintes, qui voit quoi, etc. Ces informations sont appelées des métadatas.

0.8 Bonne base de données

Une base de données peut être conçue de différentes manières. Seulement, tous les résultats possibles ne sont pas nécessairement bons. Ils peuvent notamment engendrer beaucoup de redondances, c'est-à-dire, des répétitions inutiles. C'est pourquoi une normalisation est nécessaire lors de l'étape de conception. Elle consiste à vérifier que la base de données que l'on va obtenir vérifie un certain nombre de règles.

0.9 Pourquoi est-ce qu'un étudiant futur ingénieur étudie les bases de données ?

Tout ingénieur doit maîtriser les bases de la conception des bases de données. Au cours de sa carrière, il pourra avoir à créer lui-même des bases de données. Dans d'autres cas, il pourra confier la tâche à un concepteur informaticien, mais il devra lui fournir exactement ce qu'il attend de la base de données selon ses propres besoins. Afin de pouvoir formuler cela au mieux, il devra comprendre la nature des informations à communiquer au concepteur, car un besoin mal formulé mènera probablement à la réception d'un produit inadapté.

Un ingénieur pourra avoir à gérer des bases de données de différents types : expérimentales, d'analyse, d'agrégation, d'application. Elles contiennent souvent des données de natures plus complexes que les bases de données conventionnelles [3].

Parmi les utilisations que peut faire un ingénieur d'une base de données :

— Recherche de données spécifiques ;

— Téléchargement ou transfert de données ;

— Affichage de données ;

— Analyse de données ;

— Génération de rapports ;

— Consultation de tables ou de graphes.

Ainsi, un ingénieur en maintenance industrielle d'une société devra utiliser une base de données qui regroupe l'ensemble des informations sur les équipements. Cette base de données permettra l'accès à l'information et à l'état des équipements. Un ingénieur en chimie analytique devra établir l'identité de substances chimiques inconnues. Cela se fera en utilisant des instruments adaptés et en réalisant des comparaisons avec des données présentes dans des bases des données de substances déjà identifiées.

0.10 Ce qu'apprendra l'étudiant à l'issue de ce module

À l'issue de ce module, l'étudiant se familiarisera avec les notions de base sur les bases de données et pourra :

— Modéliser un réel à travers un modèle conceptuel de données normalisé ;

— Produire un modèle logique de données relationnel prêt à être implémenté ;

— Formaliser des requêtes d'interrogation sur des relations/tables en utilisant les opérateurs de l'algèbre relationnelle ;

— Créer et interagir avec une base de données relationnelle à travers le langage SQL.

CHAPITRE 1

MODÉLISATION

1.1 Introduction

Une base de données efficace est une base de données bien conçue. Il est alors nécessaire de précéder l'étape d'implémentation de la base de données par une étape de modélisation des données.

L'étape de modélisation est ainsi l'une des étapes les plus importantes dans son cycle de vie. C'est elle qui va définir de la structure de la BDD, et c'est sur elle que reposera l'efficacité ou non de la BDD. Cette étape consiste à analyser les besoins des utilisateurs de la BDD et la nature des données à manipuler ainsi que leurs relations.

Deux modèles peuvent être produits et seront présentés dans ce chapitre : le modèle conceptuel de données (MCD) et le modèle logique de données (MLD).

Une BDD bien conçue est une BDD basée sur un modèle conceptuel qui respectera un certain nombre de règles dîtes de "normalisation". Parmi les principes les plus importants, on retrouve le principe de non-redondance et qui est le principe d'évitement de duplications (répétitions) inutiles des données dans la BDD. L'objectif principal étant d'éliminer les incohérences lors de modifications de données, car en cas de redondance, la modification d'une donnée signifie que toutes ses copies devraient être mises à jour.

1.2 Le modèle MCD

Le modèle conceptuel de données (MCD) est un modèle graphique qui a pour but de décrire les données qui seront utilisées dans la future BDD. Le MCD décrit la sémantique c'est-à-dire le sens attaché à ces données et à leurs rapports et non à l'utilisation qui peut en être faite. Cette section présente les différentes notions liées à un MCD.

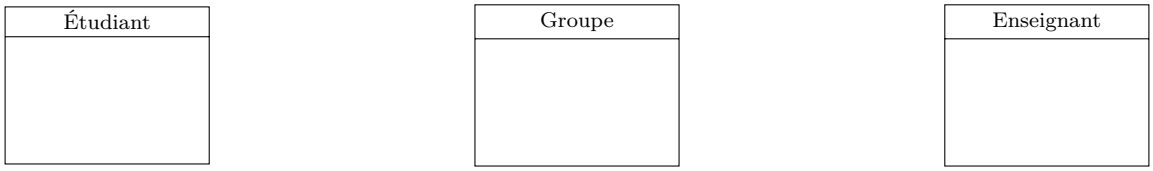
1.2.1 Entité

Définition Une entité est une classe d'objets homogènes ayant une existence propre par rapport au réel modélisé.

Exemple Exemples d'entité manipulées par le réel de la scolarité de l'école : l'entité "Étudiant", l'entité "Groupe", l'entité "Enseignent", etc.

Représentation graphique Dans un MCD, une entité est représentée par un rectangle ayant deux cases. La case supérieure contient le nom de l'entité.

Exemple



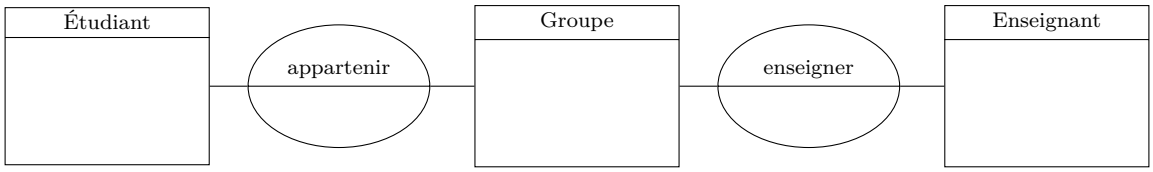
1.2.2 Association

Définition Une association est une liaison qui va lier un certain nombre d'entités dans le réel modélisé.

Exemple Dans le réel de la scolarité, il y a par exemple une association entre l'entité "Étudiant" et l'entité "Groupe", car il est important de savoir l'appartenance d'un étudiant à son groupe. Il pourrait y avoir une association entre l'entité "Étudiant" et l'entité "Enseignant", sauf qu'il serait plus intéressant qu'il y ait association entre l'entité "Groupe" et l'entité "Enseignant". La liaison entre un "Enseignant" et un "Étudiant" serait alors représentée mais de manière indirecte (Enseignant $\leftrightarrow$ Groupe $\leftrightarrow$ Étudiant).

Représentation graphique Dans un MCD, une association est représentée par une forme ovale ayant deux cases (ou une seule, voir la section "Propriété"). La case supérieure contient le nom de l'association. Le nom de l'association est souvent un verbe ou bien la concaténation des noms des entités liées par l'association. Chaque lien qui relie une association à une entité est appelée *branche*.

Exemple



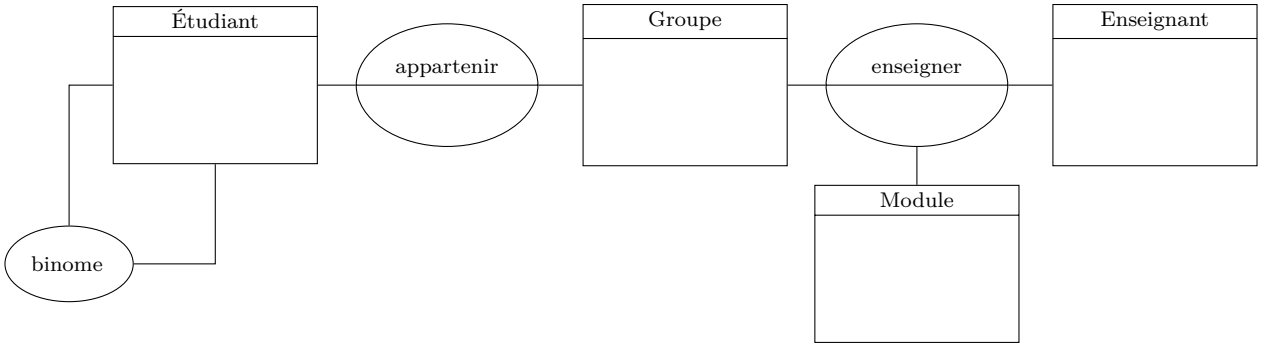
Remarques

— Le nombre de branches d'une association est dit **dimension** de cette association. Ainsi :
  — Une association à dimension 2 est dite **binaire**.
  — Une association à dimension 3 est dite **ternaire**.
  — ...
  — Une association à dimension n est dite **n-aire**.
— Une association peut lier une entité à elle même. Dans le cas où cette association possède uniquement des branches vers cette entité, elle est dite **réflexive**.

Exemples Les deux associations "appartenir" et "enseigner" de l'exemple précédent sont des associations binaires, étant donné que chacune d'elles lie exactement deux entités.

Si on considère que chaque étudiant pourrait faire partie d'un même binôme avec un autre étudiant tout au long de l'année, alors cette information pourrait être représentée par une association réflexive "binôme" qui va lier l'entité "Étudiant" à elle-même.

Si on voulait maintenant préciser, dans notre MCD, le module enseigné par un certain enseignant à un certain groupe, alors l'association binaire "Enseigner" se transformerait en association ternaire qui lierait les 3 entités "Enseignant", "Groupe" et "Module".



1.2.3 Propriété

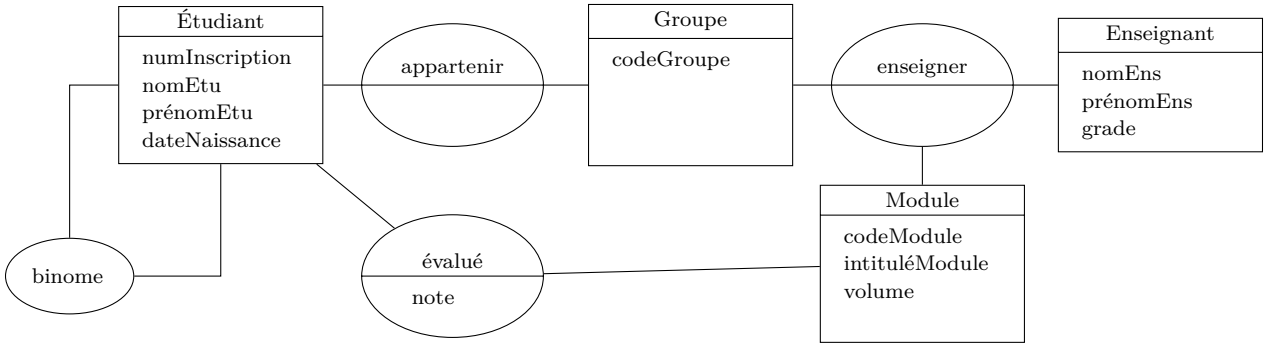
Définition Une propriété est une information simple sur une entité ou une association.

Exemple Pour un étudiant, il serait intéressant d'avoir comme propriétés : son numéro d'inscription, son nom, son/ses prénom(s), et sa date de naissance. Pour un groupe : son code. Pour un enseignant : son nom, son/ses prénom(s) et son grade. Pour un module : son code, son intitulé et son volume horaire.

Si on voulait représenter la note d'un étudiant pour chaque module, alors cela pourrait être modélisé par une association "évalué" entre l'entité "Étudiant" et l'entité "Module", la note serait alors une propriété de cette association. En effet, la note ne pourrait pas être une propriété de l'entité "Étudiant", car chaque étudiant possède plusieurs notes, et la note ne pourrait pas non plus être une propriété de l'entité "Module" car cela pourrait signifier que tous les étudiants auraient la même note pour un certain module.

Représentation graphique Les propriétés sont listées dans la case inférieure de l'entité ou de l'association concernée.

Exemple



Remarques

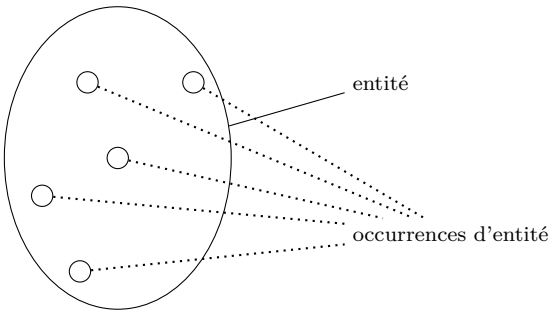
— Une entité doit posséder au moins une propriété.

— Une association peut ne pas posséder de propriétés. Dans ce cas, sa forme ovale peut être constituée d'une seule case (Exemple : association "binome" dans le MCD précédant).

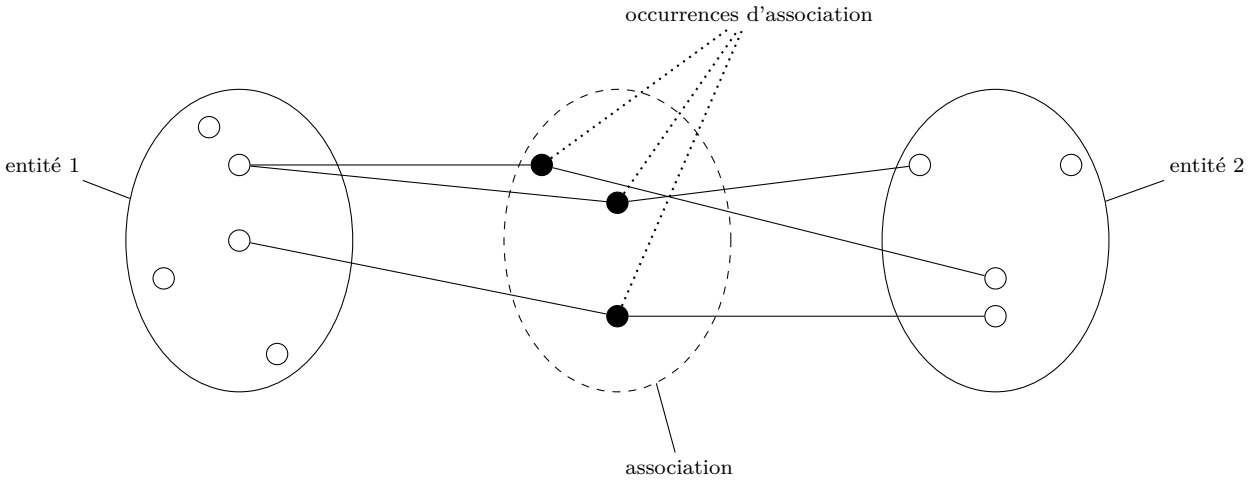
1.2.4 Occurrence

Définition Une occurrence d'une entité ou d'une association est une instanciation de cette entité ou de cette association.

Étant donné qu'une entité est une classe/famille d'objets homogènes, chaque membre de cette famille est alors une occurrence de l'entité considérée. Pour chaque occurrence d'une entité, chaque propriété prend une valeur.



L'occurrence d'une association ayant n branches, sera un lien entre un ensemble composé des n occurrences des entités liées par cette association. Ainsi, une occurrence d'une association binaire liera deux occurrences d'entités, une occurrence d'une association ternaire liera trois occurrences d'entités, etc.



Remarque Une occurrence d'une association réflexive ayant n branches liera n occurrences de la même entité.

Exemples

— Une occurrence de l'entité "Étudiant" sera un étudiant décrit à travers les valeurs de ses propriétés, ex. (numInscription=2597524, nomEtu="Bouali", prénomEtu="Ali", dateNaissance=29/02/2000).

— Une occurrence de l'entité "Groupe" sera un groupe décrit à travers ses propriétés, ex. (codeGroupe="S4").

— Si l'étudiant avec le numéro 2597524 appartient au groupe "S4", alors il y aura une occurrence de l'association "appartenir" qui sera un lien entre cette occurrence de l'entité "Étudiant" et cette occurrence de l'entité "Groupe".

1.2.5 Identifiant

Définition Un identifiant est un ensemble de propriétés (une ou plusieurs) qui permet d'identifier une occurrence d'entité ou d'association de manière unique.

Cela signifie qu'il n'existe pas deux occurrences de la même entité ou de la même association avec la même valeur d'identifiant. Toute entité ou association possède un identifiant.

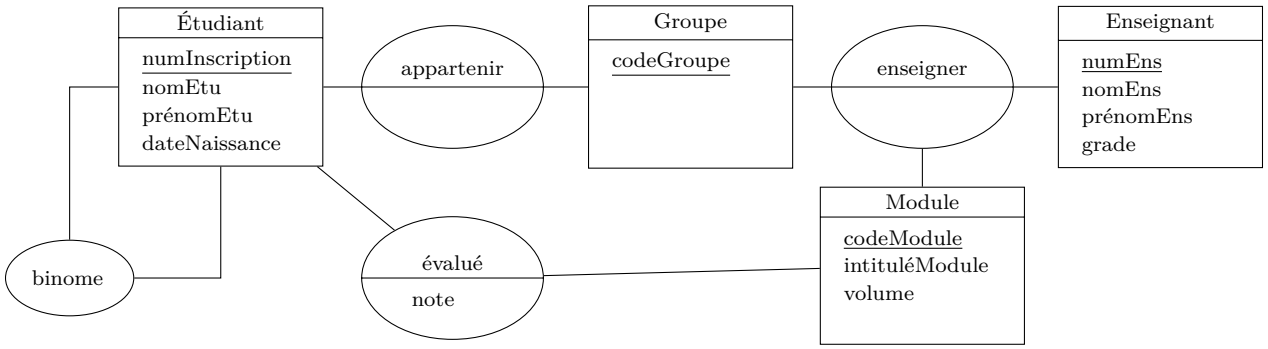
Représentation graphique

— L'identifiant d'une entité est explicite : l'ensemble des propriétés le formant sont précisées en les soulignant.

— L'identifiant d'une association est implicite : il n'est pas représenté graphiquement et est automatiquement formé par l'ensemble des identifiants des entités liées par l'association.

Exemple La propriété "numInscription" identifie de manière unique une occurrence de l'entité "Étudiant" : il n'existe pas deux étudiants avec la même valeur pour cette propriété. C'est donc un identifiant pour l'entité "Étudiant". De la même manière, "codeGroupe" est un identifiant pour l'entité "Groupe" et "codeModule" est un identifiant pour l'entité "Module". Par contre, aucune des propriétés déjà présentes dans l'entité "Enseignant" n'identifie une occurrence de l'entité de manière unique, il est alors nécessaire d'ajouter une nouvelle propriété qui jouera le rôle d'identifiant, par exemple : "numEns" qui sera un numéro unique pour chaque enseignant de l'école.

En ce qui concerne les associations, leur identifiant est implicite : on ne le représente pas graphiquement. Pour l'association "appartenir" par exemple, l'identifiant est formé automatiquement des deux propriétés "numInscription" et "codeGroupe".



Remarque Une occurrence d'association est décrite à travers les valeurs des identifiants des occurrences des entités liées, en plus des valeurs des propriétés de l'association elle-même si ces propriétés existent.

Exemple

— Si on reprend l'exemple d'occurrence d'association précédent, celui-ci est décrit à travers la valeur de l'identifiant de l'étudiant et la valeur du code du groupe (numInscription=2597524, codeGroupe="S4").

— Une occurrence de l'association "évalué" sera décrite par trois valeurs, étant donné que cette association possède une propriété, ex. (numInscription=2597524, codeModule="INF4", note=15).

1.2.6 Cardinalité

Définition Une cardinalité traduit la participation des occurrences d'une entité aux occurrences d'une association [1]. C'est un couple de valeurs (min, max) que l'on va attacher à une entité par rapport à une association. Pour identifier ces valeurs, on se pose les deux questions suivantes :

— Cardinalité minimale (min) : une occurrence quelconque de cette entité va participer à un minimum de combien d'occurrences de cette association ?

— Cardinalité maximale (max) : une occurrence quelconque de cette entité va participer à un maximum de combien d'occurrences de cette association ?

Remarque Par convention la valeur min sera mise à 0 ou bien à 1, et la valeur max sera mise à 1 ou bien à n, de la manière suivante :

— Lorsque la réponse à la question pour identifier la cardinalité minimale :

$$\begin{cases} = 0 & \text{alors} \quad min \leftarrow 0 \\ > 0 & \text{alors} \quad min \leftarrow 1 \end{cases}$$

— Lorsque la réponse à la question pour identifier la cardinalité maximale :

$$\begin{cases} = 1 & \text{alors} \quad max \leftarrow 1 \\ > 1 & \text{alors} \quad max \leftarrow n \end{cases}$$

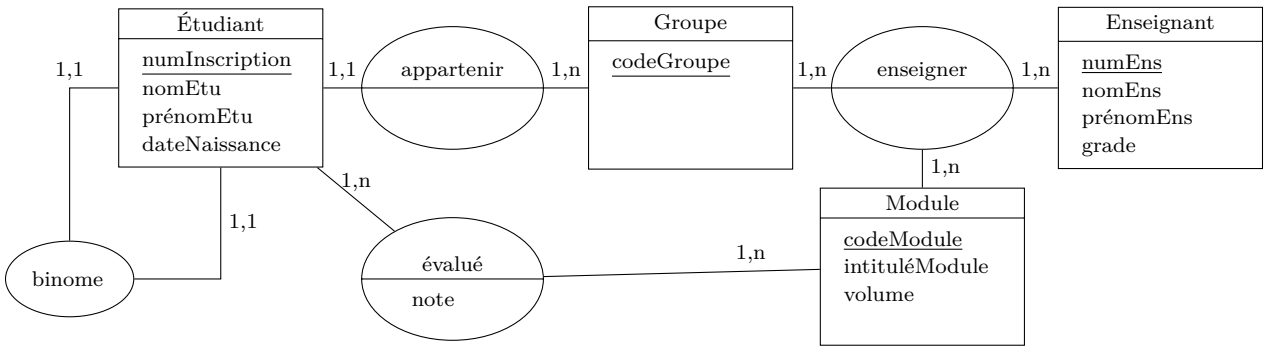
Représentation graphique Les cardinalités sont placées au niveau des branches des associations du coté des entités concernées.

Exemple Reprenons le MCD précédant :

Association	Entité	Question	En d'autres termes	Réponse	Cardinalité
appartenir (binaire)	Étudiant	Une occurrence de l'entité "Étudiant" participera à un minimum de combien d'occurrences de l'association "appartenir" ?	**Un** étudiant appartiendra à un minimum de combien de groupes ?	1	$min = 1$
		Une occurrence de l'entité "Étudiant" participera à un maximum de combien d'occurrences de l'association "appartenir" ?	**Un** étudiant appartiendra à un maximum de combien de groupes ?	1	$max = 1$
	Groupe	Une occurrence de l'entité "Groupe" participera à un minimum de combien d'occurrences de l'association "appartenir" ?	**Un** groupe sera composé d'un minimum de combien d'étudiants ?	1	$min = 1$
		Une occurrence de l'entité "Groupe" participera à un maximum de combien d'occurrences de l'association "appartenir" ?	**Un** groupe sera composé d'un maximum de combien d'étudiants ?	plusieurs (>1)	$max = n$
évalué (binaire)	Étudiant	Une occurrence de l'entité "Étudiant" participera à un minimum de combien d'occurrences de l'association "évalué" ?	**Un** étudiant sera évalué pour un minimum de combien de modules ?	1	$min = 1$
		Une occurrence de l'entité "Module" participera à un maximum de combien d'occurrences de l'association "évalué" ?	**Un** étudiant sera évalué pour un maximum de combien de modules ?	plusieurs (>1)	$max = n$
	Module	Une occurrence de l'entité "Module" participera à un minimum de combien d'occurrences de l'association "évalué" ?	**Un** module concernera l'évaluation de combien d'étudiants au minimum ?	1	$min = 1$
		Une occurrence de l'entité "Module" participera à un maximum de combien d'occurrences de l'association "évalué" ?	**Un** module concernera l'évaluation de combien d'étudiants au maximum ?	plusieurs (>1)	$max = n$

enseigner (ternaire)	Enseignant	Une occurrence de l'entité "Enseignant" participera à un minimum de combien d'occurrences de l'association "enseigner" ?	Pour **une** occurrence de l'entité "Enseignant", quel est le nombre minimal de couples d'occurrences des entités "Groupe" et "Module" auxquels il sera lié ?	1	$min = 1$
		Une occurrence de l'entité "Enseignant" participera à un maximum de combien d'occurrences de l'association "enseigner" ?	Pour **une** occurrence de l'entité "Enseignant", quel est le nombre maximal de couples d'occurrences des entités "Groupe" et "Module" auxquels il sera lié ?	plusieurs (>1)	$max = n$
	Groupe	Une occurrence de l'entité "Groupe" participera à un minimum de combien d'occurrences de l'association "enseigner" ?	Pour **une** occurrence de l'entité "Groupe", quel est le nombre minimal de couples d'occurrences des entités "Enseignant" et "Module" auxquels il sera lié ?	plusieurs (>0)	$min = 1$
		Une occurrence de l'entité "Groupe" participera à un maximum de combien d'occurrences de l'association "enseigner" ?	Pour **une** occurrence de l'entité "Groupe", quel le nombre maximal de couples d'occurrences des entités "Enseignant" et "Module" auxquels il sera lié ?	plusieurs (>1)	$max = n$
	Module	Une occurrence de l'entité "Module" participera à un minimum de combien d'occurrences de l'association "enseigner" ?	Pour **une** occurrence de l'entité "Module", quel est le nombre minimal de couples d'occurrences des entités "Enseignant" et "Groupe" auxquels il sera lié ?	plusieurs (>0)	$min = 1$
		Une occurrence de l'entité "Module" participera à un maximum de combien d'occurrences de l'association "enseigner" ?	Pour **une** occurrence de l'entité "Module", quel est le nombre maximal de couples d'occurrences des entités "Enseignant" et "Groupe" auxquels il sera lié ?	plusieurs (>1)	$max = n$
binome (réflexive)	Étudiant	Une occurrence de l'entité "Étudiant" participera à un minimum de combien d'occurrences de l'association "binome" ?	**Un** étudiant formera au minimum combien de binômes avec d'autres étudiants ?	1	$min = 1$
		Une occurrence de l'entité "Étudiant" participera à un maximum de combien d'occurrences de l'association "binome" ?	**Un** étudiant formera au maximum combien de binômes avec d'autres étudiants ?	1	$max = 1$

Remarque En ce qui concerne l'association réflexive "binome", les deux occurrences de l'entité "Étudiant" qui seront liées par une occurrence de l'association auront le même rôle, les cardinalités des deux côtés seront donc identiques. Notons que cela n'est toutefois pas applicable à toutes les associations réflexives.



1.2.7 Types d'associations binaires

Association de type fils-fils

Une association est de type fils-fils si la valeur *max* des cardinalités des deux entités liées est égale à 1. Dans ce cas, chaque occurrence de chacune des deux entités pourra être liée à une seule occurrence de l'autre entité au plus.

Association de type père-fils

Une association est de type père-fils si la valeur *max* de la cardinalité d'une des entités liées est égale à 1 alors que l'autre est égale à n. La première entité sera dite fils alors que la deuxième sera dite père, par rapport à cette association. Ainsi, une occurrence de l'entité fils ne pourra être liée qu'à une seule occurrence de l'entité père, alors qu'une occurrence de l'entité père pourra être liée à plusieurs occurrences de l'entité fils.

Association de type père-père

Une association est de type père-père si la valeur *max* des cardinalités des deux entités liées est égale à n. Dans ce cas, chaque occurrence de chacune des deux entités pourra être liée à plusieurs occurrences de l'autre entité.

Exemples Dans notre MCD :

- L'association "binome" est de type fils-fils. En effet, chaque étudiant ne pourra être relié qu'à un seul autre étudiant pour former un binôme ;
- L'association "appartenir" est de type père-fils, où l'entité "Étudiant" est le fils et l'entité "Groupe" est le père. Un étudiant n'appartiendra qu'à un seul groupe et un groupe contiendra plusieurs étudiants ;
- L'association "évalué" est de type père-père. Chaque étudiant peut être évalué pour plusieurs modules, et pour chaque module, plusieurs étudiants peuvent être évalués.

1.3 Dépendance fonctionnelle

Définition La *dépendance fonctionnelle* est une relation entre les propriétés. On dit qu'une propriété "A" dépend fonctionnellement d'une autre propriété ou d'un groupe de propriétés "B" si chaque valeur pour "B" détermine une et une seule valeur pour "A". On note B $\rightarrow$ A.

Exemple Soient les deux propriétés "numInscription" et "nomEtu" de l'entité "Étudiant".

- La propriété "nomEtu" dépend fonctionnellement de la propriété "numInscription" car pour une valeur de "numInscription" on ne peut avoir qu'une seule valeur pour "nomEtu". "numInscription" détermine donc "nomEtu". Nous avons donc numInscription $\rightarrow$ nomEtu.

— Par contre, "numInscription" ne dépend pas fonctionnellement de "nomEtu", car pour une même valeur de "nomEtu" sur plusieurs occurrences de l'entité "Étudiant", on peut avoir plusieurs valeurs pour "numInscription".

1.4 Normalisation

Normaliser un MCD permet d'obtenir une bonne BDD en assurant :

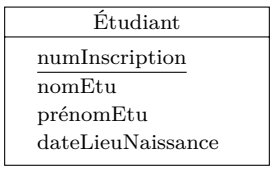
— L'évitement de redondance des données : c.à.d. éviter qu'il y ait des répétitions inutiles des données dans la BDD ;

— L'intégrité des données : c.à.d. éviter les incohérences.

Nous considérerons les règles de normalisation de base suivantes pour notre cours :

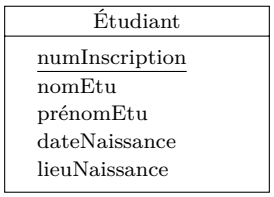
1. Chaque entité possède un identifiant ;
2. Chaque propriété est atomique, c.à.d. non décomposable et non multivaluée ;
3. Chaque propriété d'une entité qui n'appartient pas à son identifiant est en dépendance fonctionnelle directe uniquement avec cet identifiant ;
4. Chaque propriété d'une entité ou d'une association qui n'appartient pas à son identifiant est en dépendance fonctionnelle élémentaire avec cet identifiant.

1er exemple (règle 2) Soit le MCD suivant constitué uniquement de l'entité "Étudiant" dans laquelle la propriété "dateNaissance" est transformée en "dateLieuNaissance" pour contenir en même temps, la date et le lieu de naissance des étudiants.



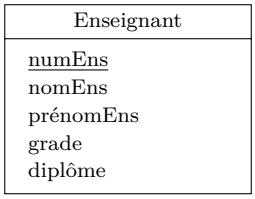
La propriété "dateLieuNaissance" n'est pas atomique car elle est décomposable : elle contient deux informations que l'on pourrait avoir besoin de manipuler séparément. Ce MCD ne respecte donc pas la 2ème règle de normalisation.

Normalisation Afin de normaliser ce MCD, il est nécessaire d'éclater la propriété "dateLieuNaissance" en deux propriétés "dateNaissance" et "lieuNaissance".



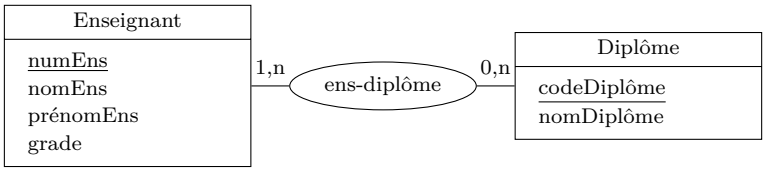
Remarque la date de naissance pourrait sembler non atomique à première vue car elle contient 3 informations : jour + mois + année. Sauf que ces 3 informations seront toujours manipulées ensemble dans cette BDD. "dateNaissance" est donc considérée comme propriété atomique. Dans un autre système, une autre date pourrait avoir besoin d'être décomposée en plusieurs propriétés selon le besoin.

2e exemple (règle 2) Soit le MCD constitué uniquement de l'entité "Enseignant" à laquelle on ajoute la propriété "diplôme" :

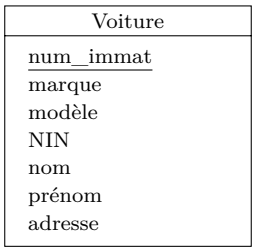


Dans ce MCD, la nouvelle propriété "diplôme" n'est pas atomique car un enseignant peut avoir plusieurs diplômes à la fois. Elle est donc multivaluée. Une même occurrence de l'entité "Enseignant" peut posséder plusieurs valeurs pour une même propriété, ce qui est en contradiction avec la 2e règle de normalisation.

Normalisation Afin de normaliser ce MCD, il est possible de considérer une nouvelle entité "Diplôme" qui sera reliée à l'entité "Enseignant" par une association. Ainsi, une occurrence d'enseignant sera reliée à autant de diplômes qu'elle possède.

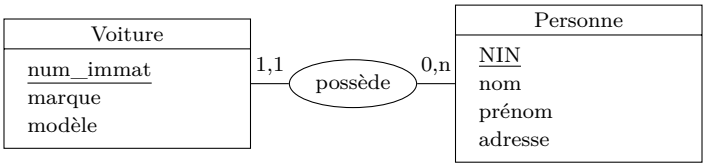


Exemple (règle 3) Soit le MCD suivant :

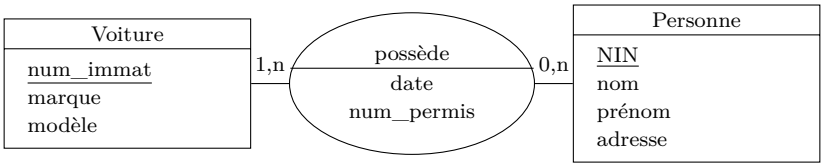


Dans cet exemple, les propriétés "nom", "prénom", et "adresse" dépendent fonctionnellement de l'attribut "NIN", i.e. numéro d'identification nationale. Elles ne dépendent pas uniquement que de "num_immat". Elles ne sont donc pas en dépendance fonctionnelle directe avec l'identifiant de l'entité "Voiture". La 3e règle de normalisation n'est donc pas respectée.

Normalisation Les propriétés "NIN", "nom", "prénom" et "adresse" constituent une entité à part entière. Elles ne doivent pas être incluses dans l'entité "Voiture" mais dans une nouvelle entité qui doit être reliée à l'entité "Voiture" par une association :

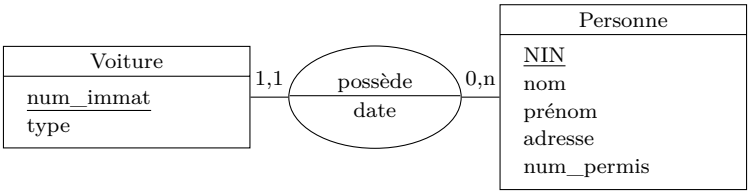


1er exemple (règle 4) Supposons qu'on modifie le MCD précédant en modifiant la cardinalité de l'entité "Voiture" à $(1, n)$ dans l'association "possède". De cette manière, nous considérons que notre base de données gardera trace des différentes "Personnes" ayant possédé la même voiture. De plus, ajoutons à l'association les deux propriétés "date" et "num_permis". "date" représente le jour de l'acquisition de la voiture par la personne et "num_permis" représente le numéro de permis de la personne.

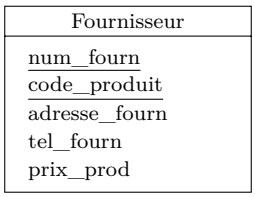


En ce qui concerne la propriété "date", celle-ci est bien en dépendance fonctionnelle élémentaire avec l'identifiant de l'association "possède" qui est formé par les deux identifiants "num_immat" et "NIN". D'un autre côté, la propriété "num_permis" dépend de l'identifiant de l'association mais il dépend également de "NIN", et donc, d'une partie de l'identifiant. Il n'est donc pas en dépendance fonctionnelle élémentaire avec l'identifiant de l'association "possède". Ce MCD ne respecte donc pas la 4e règle de normalisation.

Normalisation La propriété "num_permis" devrait apparaître comme propriété de l'entité "Personne" et non pas comme propriété de l'association "possède" :

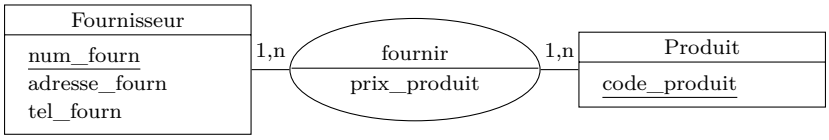


2e exemple (règle 4) Enfin, soit le MCD suivant pour une entreprise qui a besoin d'une BDD avec sa liste de fournisseurs pour chaque produit :



Dans ce MCD, la seule propriété qui n'appartient pas à l'identifiant qui est en dépendance fonctionnelle élémentaire avec l'identifiant est la propriété "prix_produit" car le prix d'un produit dépend du code du produit mais aussi du fournisseur. Quant aux propriétés "adresse_fourn" et "tel_fourn" elles dépendent également de la propriété "num_fourn" et donc d'une partie de l'identifiant. Ce MCD ne respecte donc pas la 4e règle de normalisation.

Normalisation Afin de normaliser ce MCD, il est possible de décomposer l'entité "Fournisseur" en deux entités "Fournisseur" et "Produit" et de les relier par une association "fournir" qui contiendra la propriété "prix_prod" car cette dernière dépend des identifiants des deux entités.



1.5 Le modèle MLD

Une fois le MCD construit et normalisé, nous pouvons passer la construction du modèle logique de données (MLD) d'une BDD, avant d'arriver à l'étape d'implémentation. Pour ce cours, le MLD sera basé sur le modèle relationnel et sera composé d'un ensemble de tables (appelées aussi relations, d'où le nom du modèle relationnel), qui sont des tableaux à deux dimensions.

1.5.1 Terminologie

Table Tableau à deux dimensions. Appelé aussi relation.

Tuple Chaque ligne dans une table est appelée un *n-uplet* ou *tuple*. Il ne peut pas y avoir deux tuples identiques dans une même table [4] ;

Attribut Le titre de chaque colonne dans une table est appelé *attribut* ;

Composant Une donnée à l'intersection d'une ligne et d'une colonne est appelée *composant* du tuple ;

Domaine Un *domaine* d'un attribut est la plage de valeurs qui peuvent être prises par les composants appropriés ;

Schéma de table Une table T avec une liste d'attributs $A_1, A_2, \ldots, A_m$ sera décrite par son *schéma* : $T(A_1, A_2, \ldots, A_m)$;

Schéma d'une BDD Le schéma d'une base de données est l'ensemble des schémas des tables de cette base de données ;

Clé primaire Une *clé primaire* est un attribut ou un ensemble d'attributs d'une table qui permet d'identifier les tuples de cette table de manière unique. Elle est soulignée par un trait continu ;

Clé étrangère Une *clé étrangère* est un attribut ou un ensemble d'attributs d'une table qui est à l'origine clé primaire d'une autre table. Elle permet donc de

relier les tables. Pour toute valeur de clé étrangère dans une table, il doit y avoir une valeur de clé primaire correspondante dans la table d'origine. Une clé étrangère peut être marquée par le symbole *, ou elle peut être soulignée par un trait pointillé.

1.5.2 Passage d'un MCD vers un MLD

Pour le passage d'un MCD vers un MLD, il y a lieu d'exécuter les étapes suivantes :

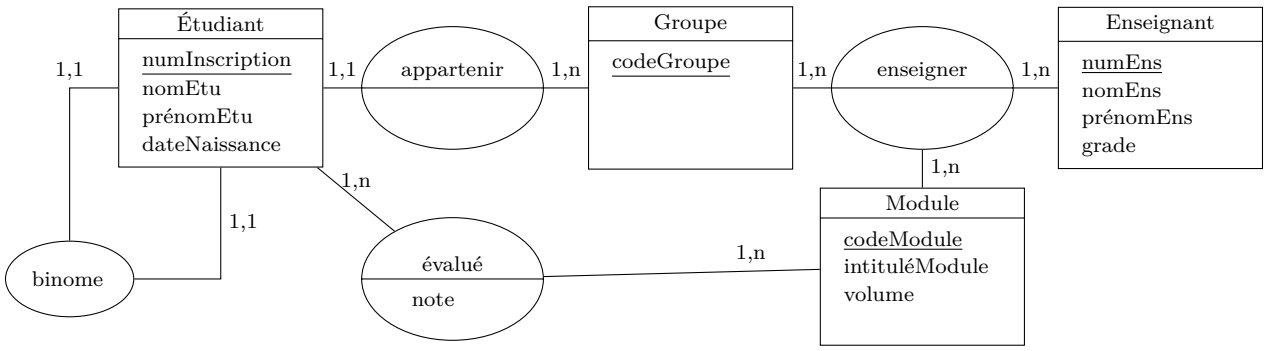
1. En ce qui concerne les entités du MCD :
   - Toute entité se transforme en table dans le MLD.
   - Ses propriétés deviennent attributs de la table dans le MLD.
   - Son identifiant est la clé primaire de la table dans le MLD.
2. En ce qui concerne les associations :
   (a) Pour une association de type père-fils :
   - L'identifiant de la table père est ajouté comme attribut à la table fils et sera clé étrangère.
   - Les propriétés de l'association sont aussi ajoutées comme attributs à la table fils.

   (b) Pour une association de type père-père :
   - L'association se transforme en table dans le MLD.
   - Les identifiants des entités liées sont attributs de la nouvelle table et joueront chacun le rôle de clé étrangère. Les deux ensemble forment la clé primaire de la table.
   - Les propriétés de l'association sont des attributs dans la nouvelle table.

   (c) Pour une association de type fils-fils : une telle association est traitée au cas par cas selon l'importance des entités liées et le pourcentage de participation des entités à l'association.

Remarque Les associations ternaires ou d'une dimension plus importante sont traitées avec le même principe que ci-dessus.

Exemple Reprenons l'exemple du MCD de la scolarité vu précédemment :



Ce MCD contient 4 entités. Le MLD correspondant contiendra donc au moins les 4 tables suivantes :

— Étudiant

— Groupe

— Enseignant

— Module

En ce qui concerne les associations :

— L'association père-fils "appartenir" : la table de l'entité fils prendra l'identifiant de son père comme clé étrangère ;

— L'association père-père "évaluer" : se transformera en une table ;

— L'association ternaire "enseigner" : peut être traitée comme une association père-père étant donné que toutes les cardinalités *max* sont à *n*. Elle se transforme donc en une nouvelle table ;

— L'association réflexive fils-fils "binome" : la table "Étudiant" prendra "numInscriptionBinome" comme clé étrangère, étant que les cardinalités *min* sont à 1 sur les deux branches de l'association (chaque étudiant formera un binôme avec un autre étudiant).

Le MLD produit est alors :

— Étudiant(numInscription, nomEtu, prénomEtu, dateNaissance, codeGroupe*, numInscriptionBinome*)

— Groupe(codeGroupe)

— Enseignant(numEns, nomEns, prénomEns, grade)

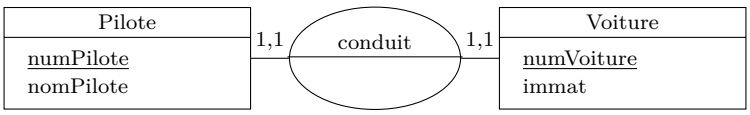
— Module(codeModule, intituléModule, volume)

— évalué(numInscription*, codeModule*, note)

— enseigner(numEns*, codeModule*, codeGroupe*)

Autres exemples pour illustrer les associations fils-fils

Exemple cas (1 : 1) - Cas d'association où le $min = 1$ pour les cardinalités des deux entités Considérons le MCD suivant concernant une compétition automobile.



Étant donné que les cardinalités *min* sont égales à 1 pour les deux entités de l'association, leur pourcentage de participation est total (il est de 100%).

Si on considère que l'une des deux entités est plus importante que la seconde, alors la première absorbera la seconde dans le MLD.

De ce fait, si on considère que l'entité "Pilote" est plus importante que l'entité "Voiture", alors "Pilote" absorbera "Voiture", ce qui donne le MLD suivant :

— Pilote(numPilote, nomPilote, numVoiture, immat)

Si on considère que l'entité "Voiture" est plus importante que l'entité "Pilote", alors "Voiture" absorbera "Pilote" ce qui donne le MLD suivant :

— Voiture(numVoiture, immat, nomPilote, numPilote)

Si les deux entités ont la même importance, alors deux MLD sont possibles. Premier MLD :

— Pilote(numPilote, nomPilote, numVoiture*)

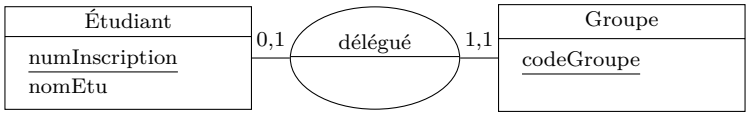
— Voiture(numVoiture, immat)

Deuxième MLD :

— Voiture(numVoiture, immat, numPilote*)

— Pilote(numPilote, nomPilote)

Exemple cas (0 : 1) - Cas d'association où $min = 0$ pour une entité et $min = 1$ pour l'autre entité Considérons le MCD représentant l'association "délegué" entre l'entité "Étudiant" et l'entité "Groupe".



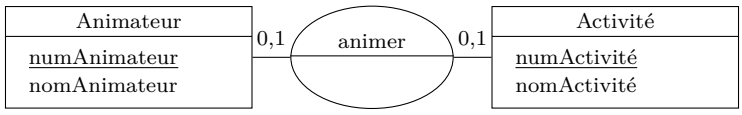
Étant donné que l'entité "Groupe" est de cardinalité 1,1 dans l'association "délégué", on est sûr que chaque occurrence de "Groupe" sera reliée à une occurrence de "Étudiant". Le pourcentage de participation de "Groupe" dans l'association "délégué" est total alors que celui de "Étudiant" est beaucoup plus faible.

La table "Groupe" prend alors "numInscription" comme clé étrangère, ce qui donne le MLD suivant :

— Étudiant(numInscription, nomEtu)

— Groupe(codeGroupe, numInscription*)

Exemple cas (0 : 0) - Cas d'association où $min = 0$ pour les cardinalités des deux entités Considérons le MCD représentant l'association "animer" entre l'entité "Animateur" et l'entité "Activité" pour modéliser le réel de l'évènement d'un club scientifique.



Ici, le pourcentage de participation d'aucune entité n'est total.

Si on considère que le pourcentage de participation de l'entité "Animateur" est élevé, alors "numActivité" est ajouté en clé étrangère à la table "Animateur". Nous obtenons le MLD :

— Animateur(numAnimateur, nomAnimateur, numActivité*)

— Activité(numActivité, nomActivité)

Si on considère que le pourcentage de participation de l'entité "Activité" est élevé, alors "numAnimateur" est ajouté en clé étrangère à la table "Activité". Nous obtenons le MLD :

— Animateur(numAnimateur, nomAnimateur)

— Activité(numActivité, nomActivité, numAnimateur*)

Si on considère que le pourcentage de participation deux deux entités est faible, alors l'association "animer" est transfrmée en table dans le MLD correspondant. Nous obtenons donc :

— Animateur(numAnimateur, nomAnimateur)

— Activité(numActivité, nomActivité)

— animer(numAnimateur*, numActivité*)

1.6 Exercices

Exercice 1

L'entreprise de vente de papier "Oran Papier" utilise le tableau 1.2 pour garder trace des différentes informations liées à ses ventes.

Les titres des différentes colonnes représentent ce qui suit :

date date de la vente

ref_ papier références du modèle de papier

format_papier format du papier

nom_c nom du client

adresse_livraison adresse de livraison si la vente nécessite une livraison

nom_v nom du vendeur

prénom_v prénom du vendeur

nb_ventes nombre de ventes du vendeur

nom_l nom du livreur si la vente nécessite une livraison

prénom_l prénom du livreur si la vente nécessite une livraison

prix_unit prix unitaire du paquet de papier

qt quantité de paquets de papier vendus

total total de la vente

Quelles remarques pouvez-vous faire par rapport à cette méthode de stockage des données ?

Tableau 1.2 – Méthode de stockage des données de l'entreprise "Oran papier".

date	ref_ papier	format_papier	nom_c	adresse_livraison	nom_v	prénom_v	nb_ventes	nom_l	prénom_l	prix_unit	qt	total
02/01/21	F1ER	A4	Aicha Bendoukha	Z.I. Hassi Ameur, Oran	Ali	Chaouche	425	Mourad	Iza	600	3	1800
02/01/21	F1ER	A4	AZ store	Rue de Flanel, Cite Ain Allah	Ali	Chaouche	426	Younes	Bouali	600	2	1100
02/01/21	GR88	A5	Ali Ali		Ali	Chaouche	325			450	4	1800
03/01/21	F1ER	A4	Bey Shop	3, Hey El Bassatine	Malika	Kaci	12	Ali	Missoum	600	20	12000
04/01/21	R58R	A3	Lamia Boukaci		Malika	Kaci	13			800	10	8000
04/01/21	R58R	A4	Malik Chikh		Malika	Kaci	14			800	20	16000
04/01/21	R58R	A4	AB com		Malika	Kaci	15			800	15	12000

Exercice 2

Mettre dans l'ordre les étapes suivantes pour la réalisation d'une base de données :

— Modélisation conceptuelle (élaborer le MCD).
— Codage de la base de données dans un langage tel que SQL pour une SGBD.
— Analyse du réel, c.à.d. de la situation existante et des besoins.
— Élaboration du modèle logique de données (MLD).

Exercice 3

1. Parmi les notions ci-dessous, quelles sont celles associées au modèle MCD ? au modèle MLD ? à aucun des deux ?

1. Clé étrangère	3. Sommet	5. Table
2. Propriété	4. Entité	6. Isthme

2. Que représente la forme ovale dans le modèle MCD ?

3. Par quoi est formé l'identifiant d'une association dans un MCD ?

4. Quelle est la question à poser afin de trouver la cardinalité *min* d'une entité par rapport à une association ?

5. Sur quoi se base-t-on pour dire qu'une association est de type "Père-Père" ?

6. Répondre par vrai ou faux :

   a) Dans un MCD, une cardinalité est un triplet de valeurs (c'est à dire : 3 valeurs).

   b) Dans un MCD, une association relie toujours 2 entités.

   c) Dans un MLD, une table peut avoir plusieurs clés primaires.

   d) Dans un MLD, une table peut avoir plusieurs clés étrangères.

Exercice 4 [1]

Un laboratoire souhaite gérer les médicaments qu'il fabrique.

Un médicament est décrit par un nom, qui permet de l'identifier, c.à.d. qu'il n'existe pas deux médicaments avec le même nom. Un médicament comporte une description courte en français, ainsi qu'une description longue en latin. On souhaite également garder trace du nombre de comprimés par boîte.

Donner le MCD qui modélise ce réel.

1. https ://stph.scenari-community.org/bdd/0/co/rel4e101.html

Exercice 5

On souhaite modéliser la gestion d'une base de données musicale composée d'un ensemble d'albums. Nous savons que :

— Un album est identifié par un numéro unique

— Un album possède un titre

— Un album peut contenir plusieurs œuvres (une œuvre pouvant être une musique, une chanson, ...)

— Une œuvre peut être associée à un ou plusieurs artistes, ou à un groupe (un groupe étant un ensemble d'artistes)

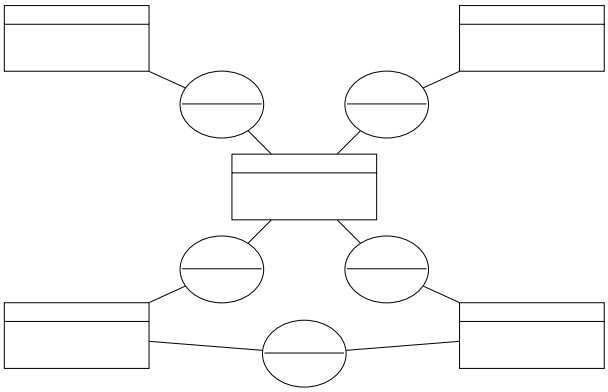
— Les œuvres dans un album sont classées et un numéro d'ordre permet de repérer une œuvre dans un album

— Les œuvres sont regroupées par style (chaâbi, rock, ...)

— Une œuvre peut-être disponible sur plusieurs albums

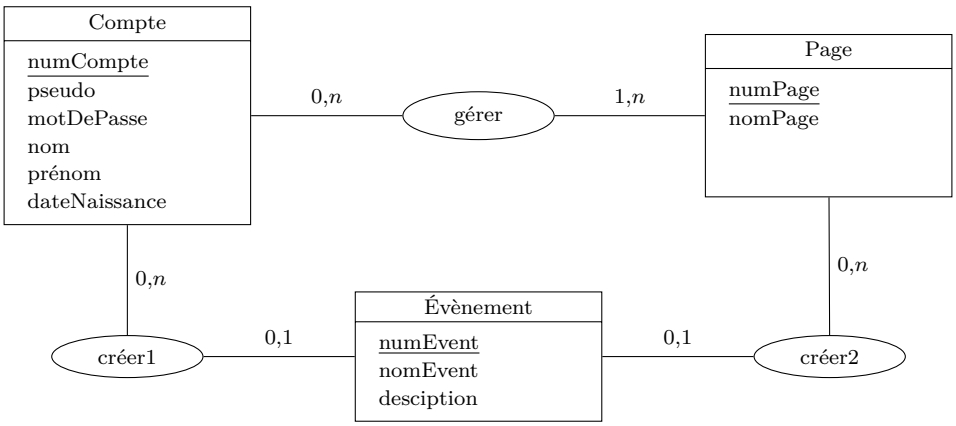
— Une œuvre peut être un mélange de styles

Compléter le MCD suivant :



Exercice 6

Soit le MCD suivant qui modélise un réseau social virtuel :

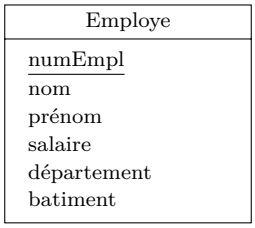


En se basant sur ce MCD, répondre aux questions suivantes en justifiant :

1. Deux comptes peuvent-ils avoir le même mot de passe ?
2. Une page peut-elle n'être gérée par aucun compte ?
3. Un évènement peut-il ne pas avoir de créateur ?

Exercice 7

Soit le MCD suivant :

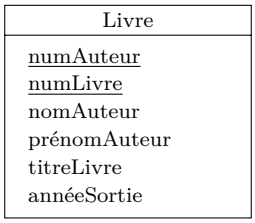


Sachant qu'un employé travaille dans un département donné, et qu'aucun département ne possède des locaux dans plusieurs bâtiments.

1. Quelles sont les dépendances fonctionnelles de ce MCD ?
2. Ce MCD est-il normalisé ? Justifier.
3. Si la réponse est non, le normaliser.

Exercice 8

Soit le MCD suivant :



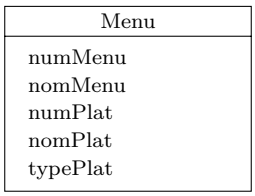
La propriété "numLivre" indique le numéro d'un livre par rapport à un auteur précis.

1. Quelles sont les dépendances fonctionnelles de ce MCD ?
2. Ce MCD est-il normalisé ? Justifier.
3. Si la réponse est non, le normaliser.

Exercice 9

Considérons un restaurant avec plusieurs menus, et dans chaque menu, plusieurs plats. Un plat pouvant apparaître dans plusieurs menus.

Soit le MCD suivant constitué d'une seule entité :



1. Donner les dépendances fonctionnelles de ce MCD.

2. Proposer un identifiant pour cette entité.

3. En considérant cet identifiant, quelle(s) règle(s) de normalisation vue(s) en cours n'est/ne sont pas respectée(s) dans ce MCD ?

4. Proposer une normalisation pour ce MCD.

5. Donner le MLD correspondant à ce MCD normalisé.

Exercice 10

Une entreprise organise un salon de tourisme. Lors de ce salon, un exposant peut louer un ou plusieurs stands, c.à.d. des espaces. Chaque stand possède un numéro et une superficie.

Chaque exposant possède un numéro, un nom et un numéro de téléphone. Il peut appartenir à un ou plusieurs domaines (hôtellerie, restauration, etc.). Un exposant peut être de type entreprise ou bien de type club.

Le salon dure 4 jours, mais un exposant peut louer un stand uniquement durant les jours qui l'arrangent.

Modéliser ce réel par un MCD puis donner le MLD correspondants.

Exercice 11 [2]

L'entreprise ZéroDéfo veut répertorier ses fautes de production. L'entreprise veut associer les fautes aux produits concernés. Chaque faute est classifiée dans des catégories. Chaque produit est basé sur un modèle.

2. https ://stph.scenari-community.org/bdd/0/co/revUE003.html

— Pour chaque modèle, on veut gérer son code constitué de 8 caractères alphanumériques, son nom et la date de mise sur le marché.

— Pour chaque produit, on veut connaître le modèle associé, le numéro de série et le numéro de produit ainsi que l'année de production. Un produit est identifié par son numéro de série et son numéro de produit.

Plusieurs produits partagent le même numéro de série (tous les produits de cette série), et deux produits peuvent avoir (par hasard) le même numéro de produit, dans des séries différentes (qui ont adopté le même système de codage des produits).

— Une faute concerne toujours un produit. Elle possède un code unique, un titre et la date de détection. Elle peut éventuellement avoir un commentaire et la date de réparation si le produit a été réparé.

— Une faute est toujours classifiée dans une catégorie au moins (elle peut être classifiée dans plusieurs).

— Les catégories possèdent un nom et, optionnellement, une description.

Donner le MCD et le MLD correspondants.

Exercice 12

Soit la base de données avec le MLD suivant :

— Client(num_client, nom, prenom, ville)

— Compte(num_compte, num_client*, type)

— Operation(num_op, num_compte*, montant, informations)

Donner un MCD étant à l'origine de ce MLD.

Exercice 13

On considère une base de données qu'on va nommer BDD et qui va contenir les informations d'un certain nombre de bases de données. Avant d'implémenter BDD, nous devons avoir son MCD qui est constitué des entités suivantes :

Nom de l'entité	Propriété de l'entité
Modèle	id, nom
Entité	id, nom
Association	id, nom
Propriété	id, nom, type

La propriété "id" est l'identifiant de chaque entité.

1. Si on considère l'association "entité_possède_propriété", quelles seront les cardinalités de cette association ? Justifier.

2. Si on considère l'association "association_possède_propriété", quelles seront les cardinalités de cette association ? Justifier.

3. Si on considère l'association "association_relie_entités", est-ce que cette association possède des propriétés ? Justifier.

1.7 Solutions des exercices

Solution de l'exercice 1

Cette méthode de représentation représente beaucoup d'inconvénients parmi lesquelles il est possible de citer :

— Il y a des informations sur trop d'objets dans un même tableau (vente, papier, client, vendeur, livreur).

— Pour chaque objet, ex. chaque vendeur, il y a les mêmes informations qui se répètent plusieurs fois (redondance).

— La redondance qui cause des incohérences : la référence R58R qui est parfois du format A3 et d'autres fois du A4.

— Perte d'espace dans le cas de ventes sans livraison pour les colonnes adresse_livraison, nom_l et prénom_l.

— Colonne "totale" stockée pour chaque vente alors que l'information peut être déduite d'autres colonnes, i.e. "prix_unit" et "qt" (forme de redondance).

— On ne sait pas s'il y a un seul "Ali Chaouche" ou deux. Pas d'information pouvant le désigner de manière sûre.

— La colonne "nb_vente" n'est pas claire : est-ce qu'elle représente le nombre total de ventes du vendeur ou bien si celui-ci est attaché à la vente en question.

— ...

Solution de l'exercice 2

1. Analyse du réel, c.à.d. de la situation existante et des besoins.
2. Modélisation conceptuelle (élaborer le MCD).
3. Élaboration du modèle logique de données (MLD).
4. Codage de la base de données dans un langage tel que SQL pour une SGBD particulier.

Solution de l'exercice 3

1. — Notions associées au modèle MCD : Propriété, Entité.
   — Notions associées au modèle MLD : Clé primaire, Table.
   — Notions non associées au modèle MCD ni au modèle MLD : Sommet, Isthme.

2. Dans un MCD, la forme ovale représente une association.

3. L'identifiant d'une association dans un MCD est formé par la concaténation des identifiants des entités liées par l'association.

4. La question à poser afin de trouver la cardinalité *min* d'une entité par rapport à une association est : chaque occurrence de cette entité participera au minimum à combien d'occurrences de cette association.

5. Une association est dite de type père-père si la cardinalité *max* au niveau des deux entités reliées par l'association est n (supérieure à 1).

6. a) Faux. Dans un MCD, une cardinalité est un couple de valeurs (c'est à dire : 2 valeurs).

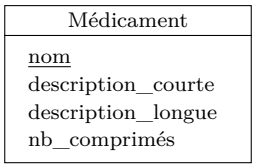
   b) Faux. Dans un MCD, une association peut relier une, deux ou plus d'entités.

   c) Faux. Dans un MLD, une table possède toujours une seul clé primaire.

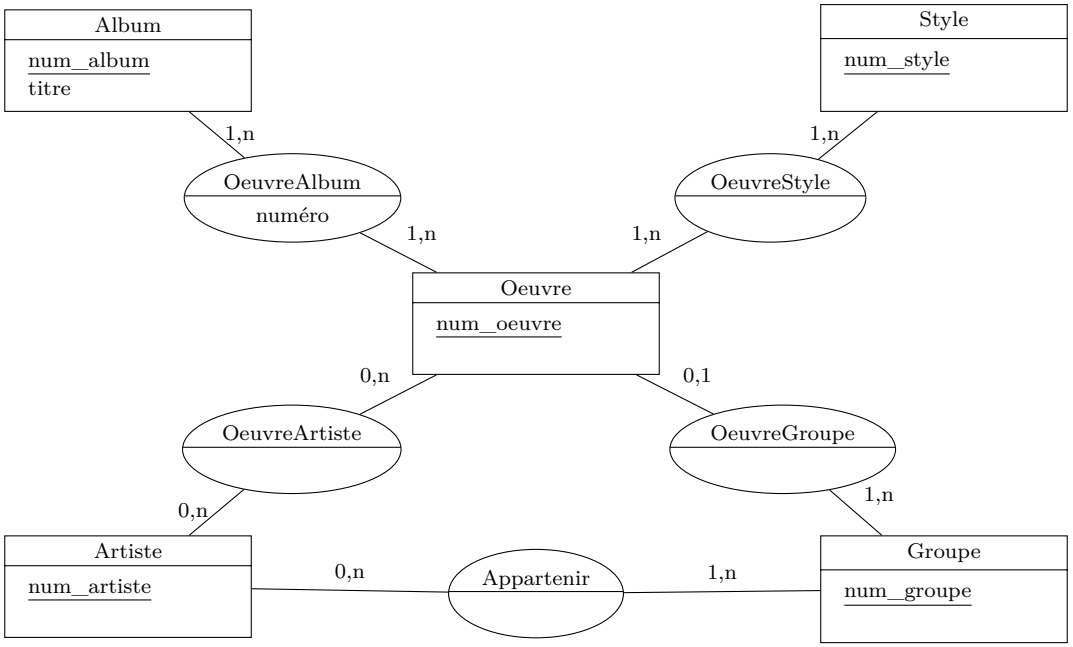
   d) Vrai. Dans un MLD, une table peut avoir plusieurs clés étrangères.

Solution de l'exercice 4

Le réel donné peut être modélisé par un MCD à unique entité comme suit :



Solution de l'exercice 5

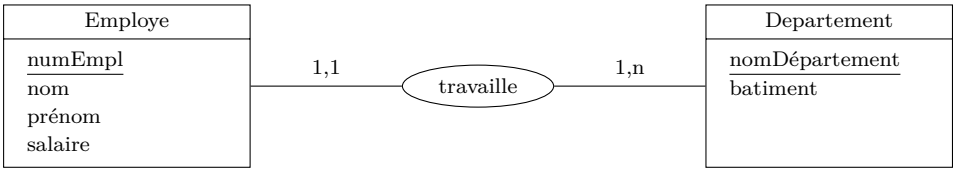


Solution de l'exercice 6

1. Oui, car d'après ce MCD, rien n'empêche deux comptes d'avoir le même mot de passe. Deux occurrences d'une même entité peuvent avoir la même valeur pour une même propriété du moment qu'il ne s'agit pas de l'identifiant.
2. Non, car d'après ce MCD, une page doit nécessairement être gérée par au moins un compte à cause de la cardinalité $min = 1$ de l'association "gérer" du coté de l'entité "Page". Une occurrence de "page" doit participer au minimum à une occurrence de "gérer".
3. Oui, d'après ce MCD, un évènement peut n'avoir aucun créateur, à cause des cardinalités $min = 0$ pour les associations "créer1" et "créer2" du coté de l'entité "Évènement". Une occurrence de "Évènement" peut ne participer à aucune occurrence des deux associations.

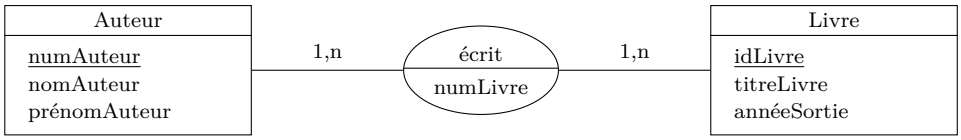
Solution de l'exercice 7

1. Parmi les dépendances fonctionnelles de ce MCD nous avons :
   - numEmpl → nom
   - numEmpl → prénom
   - numEmpl → salaire
   - numEmpl → département
   - numEmpl → batiment
   - département → batiment
2. Ce MCD n'est pas normalisé car la propriété "batiment" n'est pas en dépendance fonctionnelle directe uniquement avec l'identifiant de l'entité. Elle dépend également d'une autre propriété qui est "département". Ce MCD ne respect donc pas la 3e règle de normalisation vue en cours.
3. La normalisation de ce MCD peut se faire en créant une deuxième entité "département" comme suit :



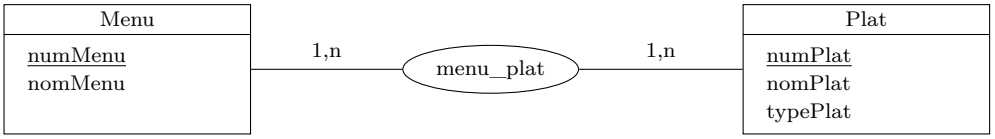
Solution de l'exercice 8

1. Parmi les dépendances fonctionnelles de ce MCD nous avons :
   - numAuteur → nomAuteur
   - numAuteur → prénomAuteur
   - numAuteur,numLivre → titreLivre
   - numAuteur,numLivre → annéeSortie
2. Ce MCD n'est pas normalisé car les propriétés "nomAuteur" et "prénomAuteur" ne sont pas en dépendance fonctionnelle élémentaire avec l'identifiant de l'entité. Elles dépendent d'une partie de l'identifiant également, i.e. de la propriété "numAuteur". Ce MCD ne respecte donc pas la 4e règle de normalisation vue en cours.
3. La normalisation de ce MCD peut se faire en créant une deuxième entité "Auteur" comme suit :



Solution de l'exercice 9

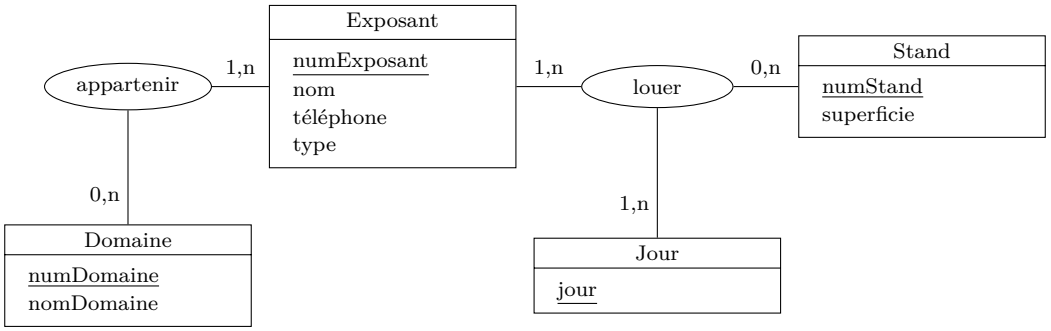
1. Parmi les dépendances fonctionnelles de ce MCD nous avons :
   — numMenu → nomMenu
   — numPlat → nomPlat
   — numPlat → typePlat
2. L'identifiant est composé des deux propriétés : numMenu, numPlat.
3. Ce MCD ne respecte par la 4e règle de normalisation : les propriétés nomMenu, nomPlat et typePlat ne sont pas en dépendance fonctionnelle élémentaire avec l'identifiant de l'entité Menu.
4. Normalisation du MCD :



5. Le MLD :
   — Menu(numMenu,nomMenu)
   — Plat(numPlat,nomPlat,typePlat)
   — travaille(numMenu*,numPlat*)

Solution de l'exercice 10

Le MCD :

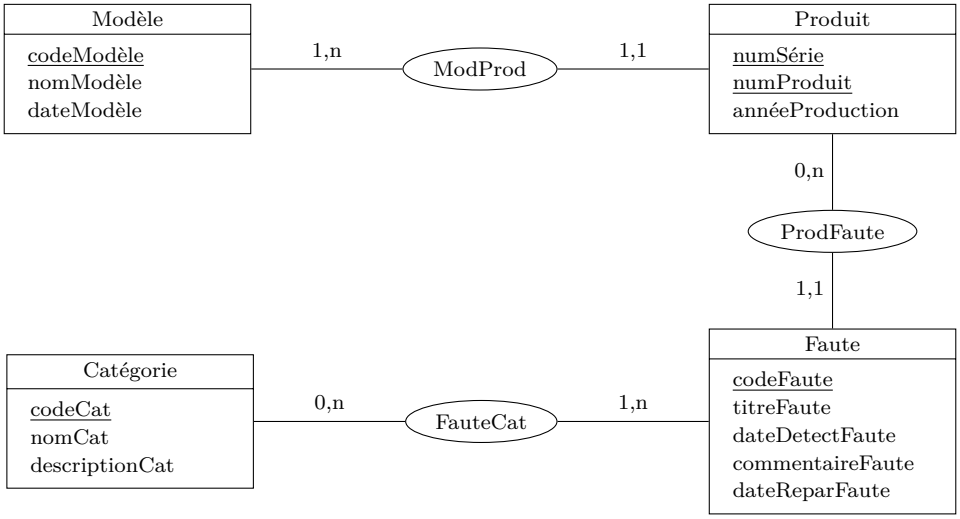


Le MLD :

— Exposant(numExposant,nom,téléphone,type)
— Domaine(numDomaine,nomDomaine)
— Stand(numStand,superficie)
— Jour(jour)
— appartenir(numDomaine*,numExposant*)
— louer(numExposant*,numStand*,jour*)

Solution de l'exercice 11

Le MCD :



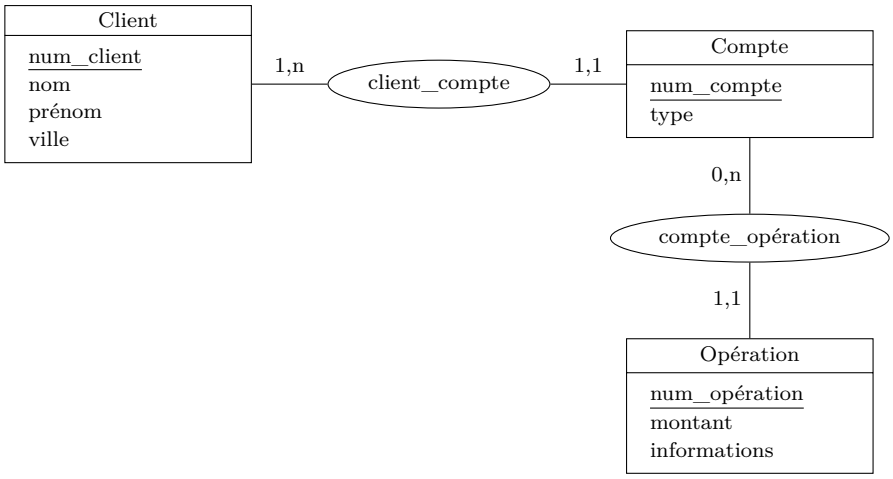
Le MLD :

— Modèle(codeModèle, nomModèle, dateModèle)
— Produit(numSérie, numProduit, annéeProduction, codeModèle)
— Faute(codeFaute, titreFaute, dateDetectFaute, commentaireFaute, dateReparFaute, numSérie, numProduit)
— Catégorie(codeCat, nomCat, descriptionCat)
— FauteCat(codeFaute, codeCat)

Solution de l'exercice 12

— Client(num_client, nom, prenom, ville)
— Compte(num_compte, num_client*, type)
— Operation(num_op, num_compte*, montant, informations)

Un MCD étant à l'origine du MLD ci-dessus :



Solution de l'exercice 13

1. Si on considère l'association "entité_possède_propriété", les cardinalités de cette association seront :
   — Coté "entité" : la cardinalité sera (1,n) car chaque entité doit possèder au moins une propriété.
   — Coté "propriété" : la cardinalité sera (0,1) car chaque propriété appartiendra soit à une entité ou bien à une association. (en considérant que deux propriétés ayant le même nom et appartenant à deux entités/associations sont deux propriétés différentes)
2. Si on considère l'association "association_possède_propriété", les cardinalités de cette association seront :
   — Coté "association" : la cardinalité sera (0,n) car une association peut ne posséder aucune propriété, comme elle peut en posséder plusieurs.
   — Coté "propriété" : la cardinalité sera (0,1) car comme précédemment chaque propriété appartiendra soit à une entité ou bien à une association. (en considérant que deux propriétés ayant le même nom et appartenant à deux entités/associations sont deux propriétés différentes)
3. Si on considère l'association "association_relie_entités", celle-ci possèdera comme propriétés les cardinalité min et max de l'entité concernée par rapport à l'association concernée.

CHAPITRE 2

L'ALGÈBRE RELATIONNELLE

2.1 Introduction

L'algèbre relationnelle est un langage de requêtes sur les bases de données relationnelles. Elle permet de construire de nouvelles relations à travers des opérations sur une ou plusieurs autres relations. C'est ce qui nous permet d'effectuer des recherches d'informations précises dans une base de données [5].

L'algèbre relationnelle est la base sur laquelle repose des langages de gestion de bases de données tels que SQL.

Pour la suite de ce chapitre, les exemples donnés seront illustrés en considérant la base de données relationnelle de gestion des nouveaux bacheliers avec le MLD suivant :

— Établissement(numEta, nomEta, ville)
— Bachelier(numBach, nom, prénom, ville, moyenne)
— Filière(numFilière, intitulé)
— Postuler(numBach*, numEta*, numFilière*)

et les relations (tables) suivantes :

Relation Établissement

numEta	nomEta	ville
01	Univ Alger 1	Alger
02	ESGEE Oran	Oran
03	Univ Oran 1	Oran

Relation Bachelier

numBach	nom	prénom	ville	moyenne
01	Belaid	Houda	Mascara	15,35
02	Soufi	Ali	Meliana	10,01
03	Berrahou	Malika	Oran	12,00
04	Maaziz	Houda	Tlemcen	11,58

Relation Filière

numFilière	intitulé
01	Architecure
02	MI
03	Prépa ST
04	ST
05	Droit

Relation Postuler

numBach	numEta	numFilière
01	01	01
01	02	03
01	03	04
02	01	05
03	01	05
03	03	02

2.2 Opérateurs de l'algèbre relationnelle

Parmi les opérateurs de l'algèbre relationnelle existants, nous traiterons : la sélection, la projection, le renommage, l'union, l'intersection, la différence, le produit cartésien, la jointure, la jointure naturelle, et la division.

2.2.1 La sélection

Notée $\sigma_E(R)$, la sélection est appliquée à une relation R et une expression logique E. Elle donne en résultat une nouvelle relation qui possède le même schéma que R mais dont les tuples sont uniquement ceux qui satisfont l'expression E.

Exemples :
Pour obtenir tous les bacheliers dont la moyenne est supérieure ou égale à 12, on applique :

$$R_1 = \sigma_{\text{moyenne} \geq 12}(\text{Bachelier})$$

Relation R_1

numBach	nom	prénom	ville	moyenne
01	Belaid	Houda	Mascara	15,35
03	Berrahou	Malika	Oran	12,00

Pour obtenir tous les étudiantes dont le prénom est "Houda" et dont la moyenne est inférieure à 13 on applique :

$$R_2 = \sigma_{\text{prénom='Houda'} \wedge \text{moyenne} < 13}(\text{Bachelier})$$

Relation R_2

numBach	nom	prénom	ville	moyenne
04	Maaziz	Houda	Tlemcen	11,58

2.2.2 La projection

Notée $\pi_{A_1,A_2 \ldots A_n}(R)$, la projection est appliquée à une relation R et un ensemble d'attributs $A_1, A_2 \ldots A_n$. Elle donne en résultat une nouvelle relation dont les attributs autres que $A_1, A_2 \ldots A_n$ sont supprimés. Les tuples doublons sont éliminés.

Exemples :
Si on souhaite obtenir uniquement le nom et la ville de tous les établissements, on appliquera l'opérateur de projection sur la relation "Établissement" de la manière suivante :

$$R_3 = \pi_{\text{nomEta,ville}}\Big(\text{Établissement}\Big)$$

Relation R_3

nomEta	ville
Univ Alger 1	Alger
ESGEE Oran	Oran
Univ Oran 1	Oran

En appliquant $R_4 = \pi_{\text{ville}}\Big(\text{Établissement}\Big)$, on peut remarquer qu'on a comme résultat toutes les villes qui apparaissent dans la relation "Établissement" et que la ville "Oran" ne se répète pas, i.e. le doublon a été supprimé.

Relation R_4

ville
Alger
Oran

Il est possible de combiner les opérateurs. De ce fait, si on veut obtenir le numéro et le nom des bacheliers avec une moyenne supérieure à 11 on appliquera :

$$R_5 = \pi_{\text{numBach,nom}}\left(\sigma_{\text{moyenne}>11}\left(\text{Bachelier}\right)\right)$$

Relation R_5

numBach	nom
01	Belaid
03	Berrahou
04	Maaziz

2.2.3 Le renommage

Noté $\rho_{[A_1:B_1,A_2:B_2,\ldots,A_n:B_n]}(R)$, le renommage est appliqué à une relation R et un ensemble de couples d'attributs (A_i, B_i), tel que A_i est attribut de R. Il donne en résultat une nouvelle relation avec le schéma de R modifié tel que chaque attribut A_i est renommé B_i. La nouvelle relation possède les mêmes tuples que R.

Exemple :
On peut obtenir la nouvelle relation R_6 à partir de la relation "Établissement" en renommant l'attribut "numEta" en "numéro" en appliquant :

$$R_6 = \rho_{[\text{numEta : numéro}]} \Big(\text{Établissement}\Big)$$

Relation R_6

numéro	nomEta	ville
01	Univ Alger 1	Alger
02	ESGEE Oran	Oran
03	Univ Oran 1	Oran

2.2.4 L'union

Notée $R_a \cup R_b$, l'union est appliquée à deux relations de même schéma R_a et R_b. Elle donne une nouvelle relation de même schéma contenant les tuples qui apparaissent dans R_a ou dans R_b. Les tuples doublons sont éliminés.

Exemple :
On peut obtenir la liste des villes qui apparaissent dans la relation "Bachelier" ou la relation "Établissement" en appliquant :

$$R_7 = \pi_{\text{ville}} (\text{Bachelier}) \cup \pi_{\text{ville}} \Big(\text{Établissement}\Big)$$

Relation R_7

ville
Mascara
Meliana
Oran
Tlemcen
Alger

2.2.5 L'intersection

Notée $R' = R_a \cap R_b$, l'intersection est appliquée à deux relations de même schéma R_a et R_b. Elle donne une nouvelle relation de même schéma contenant les tuples qui apparaissent dans R_a et dans R_b.

Exemple :
On peut obtenir la liste des villes qui apparaissent dans la relation "Bachelier" et dans la relation "Établissement" en appliquant :

$$R_8 = \pi_{\text{ville}} (\text{Bachelier}) \cap \pi_{\text{ville}} \Big(\text{Établissement}\Big)$$

Relation R_8

ville
Oran

2.2.6 La différence

Noté $R_a - R_b$, la différence est appliquée à deux relations de même schéma R_a et R_b. Elle donne une nouvelle relation de même schéma, contenant les tuples qui apparaissent dans R_a mais qui n'apparaissent pas dans R_b.

Exemple :
On peut obtenir la liste des villes qui apparaissent dans la relation "Bachelier" mais qui n'apparaissent pas dans la relation "Établissement" en appliquant :

$$R_9 = \pi_{\text{ville}}(\text{Bachelier}) - \pi_{\text{ville}}\Big(\text{Établissement}\Big)$$

Relation R_9

ville
Mascara
Meliana
Tlemcen

2.2.7 Le produit cartésien

Noté $R_a \times R_b$, le produit cartésien est appliqué à deux relations R_a et R_b. Il donne une nouvelle relation dont le schéma est la concaténation des schémas des deux relations R_a et R_b et dont les tuples sont toutes les combinaisons possibles des tuples de R_a et ceux de R_b.

Remarque : Si R_a et R_b possèdent un nom d'attribut commun A, celui-ci apparaîtra deux fois dans la nouvelle relation obtenue mais sera renommé $R_a.A$ pour désigner celui qui appartenait à R_a et $R_b.A$ pour désigner celui qui appartenait à R_b.

Exemples : Le produit cartésien $R_{10} = \text{Bachelier} \times \text{Établissement}$ donne :

Relation R_{10}

numBach	nom	prénom	Bachelier.ville	moyenne	numEta	nomEta	Établissement.ville
01	Belaid	Houda	Mascara	15,35	01	Univ Alger 1	Alger
01	Belaid	Houda	Mascara	15,35	02	ESGEE Oran	Oran
01	Belaid	Houda	Mascara	15,35	03	Univ Oran 1	Oran
02	Soufi	Ali	Meliana	10,01	01	Univ Alger 1	Alger
02	Soufi	Ali	Meliana	10,01	02	ESGEE Oran	Oran
02	Soufi	Ali	Meliana	10,01	03	Univ Oran 1	Oran
03	Berrahou	Malika	Oran	12,00	01	Univ Alger 1	Alger
03	Berrahou	Malika	Oran	12,00	02	ESGEE Oran	Oran
03	Berrahou	Malika	Oran	12,00	03	Univ Oran 1	Oran
04	Maaziz	Houda	Tlemcen	11,58	01	Univ Alger 1	Alger
04	Maaziz	Houda	Tlemcen	11,58	02	ESGEE Oran	Oran
04	Maaziz	Houda	Tlemcen	11,58	03	Univ Oran 1	Oran

On peut remarquer que l'attribut commun entre les deux relations "Bachelier" et "Établissement" est l'attribut "ville". Il a été renommé dans R_{11} respectivement en "Bachelier.ville" et "Établissement.ville".

Pour obtenir uniquement les combinaisons des tuples de la relation "Bachelier" avec ceux de la relation "Établissement" ayant la même ville, on peut appliquer :

$$R_{11} = \sigma_{\text{Bachelier.ville=Établissement.ville}} \Big(\text{Bachelier} \times \text{Établissement}\Big)$$

Ce qui donnerait :

Relation R_{11}

numBach	nom	prénom	Bachelier.ville	moyenne	numEta	nomEta	Établissement.ville
03	Berrahou	Malika	Oran	12,00	02	ESGEE Oran	Oran
03	Berrahou	Malika	Oran	12,00	03	Univ Oran 1	Oran

Afin d'obtenir le nom, le prénom et la moyenne des bacheliers qui ont une moyenne supérieure ou égale à 12 et qui ont postulé à la filière n°5, on peut appliquer :

$$R_{12} = \pi_{\text{nom,prénom,moyenne}}(\sigma_{\text{(Bachelier.numBach=Postuler.numBach)}\wedge\text{(moyenne}\geq 12)\wedge\text{(numFilière=05)}} (\text{Bachelier} \times \text{Postuler}))$$

Ce qui donnerait :

Relation R_{12}

nom	prénom	moyenne
Berrahou	Malika	12,00

2.2.8 La jointure

Notée $R_a \bowtie_E R_b$, la jointure est appliquée à deux relations R_a et R_b et une expression logique E. Elle donne une nouvelle relation dont le schéma est la concaténation des schémas des deux relations R_a et R_b et dont les tuples sont les combinaisons des tuples de R_a et ceux de R_b qui satisfont l'expression E.

Remarque : L'application de la jointure est équivalent à la combinaison d'un produit cartésien et d'une sélection, i.e. l'opération $R_a \bowtie_E R_b$ pourrait également être écrite sous la forme : $\sigma_E (R_a \times R_b)$.

Exemples : En utilisant la jointure, la relation R_{11} précédente aurait pu être écrite comme suit :

$$R_{11} = \text{Bachelier} \bowtie_{\text{Bachelier.ville=Établissement.ville}} \text{Établissement}$$

De la même manière, la relation R_{12} aurait pu être obtenue comme suit :

$$R_{12} = \pi_{\text{nom,prénom,moyenne}} \left(\text{Bachelier} \bowtie_{\text{(Bachelier.numBach=Postuler.numBach)}\wedge\text{(moyenne}\geq 12)\wedge\text{(numFilière=05)}} \text{Postuler}\right)$$

Pour obtenir les combinaisons des établissements de même ville, on pourrait appliquer :

$$R_{13} = \Big(\text{Établissement}\Big) \bowtie_{\text{ville = ville1}} \Big(\rho_{\text{[numEta :numEta1,nomEta :nomEta1,ville :ville1]}}\text{Établissement}\Big)$$

Ce qui donnerait :

Relation R_{13}

numEta	nomEta	ville	numEta1	nomEta1	ville1
01	Univ Alger 1	Alger	01	Univ Alger 1	Alger
02	ESGEE Oran	Oran	02	ESGEE Oran	Oran
02	ESGEE Oran	Oran	03	Univ Oran 1	Oran
03	Univ Oran 1	Oran	02	ESGEE Oran	Oran
03	Univ Oran 1	Oran	03	Univ Oran 1	Oran

On peut remarquer que certaines des combinaisons associent un établissement avec lui même. Pour éviter cela, on pourrait appliquer :

$$R_{14} = \Big(\text{Établissement}\Big) \bowtie_{\text{(ville = ville)}\wedge\text{(numEta}\neq\text{numEta1)}} \Big(\rho_{\text{[numEta :numEta1,nomEta :nomEta1,ville :ville1]}}\text{Établissement}\Big)$$

Ce qui donnerait :

Relation R_{14}

numEta	nomEta	ville	numEta1	nomEta1	ville1
02	ESGEE Oran	Oran	03	Univ Oran 1	Oran
03	Univ Oran 1	Oran	02	ESGEE Oran	Oran

Enfin, afin d'éviter d'avoir comme résultat des combinaisons de type (A, B) et (B, A) comme c'est le cas dans la relation R_{14}, on pourrait appliquer :

$$R_{15} = \Big(\text{Établissement}\Big) \bowtie_{\text{(ville = ville1)}\wedge\text{(numEta}<\text{numEta1)}} \Big(\rho_{\text{[numEta :numEta1,nomEta :nomEta1,ville :ville1]}}\text{Établissement}\Big)$$

Ce qui donnerait :

Relation R_{15}

numEta	nomEta	ville	numEta1	nomEta1	ville1
02	ESGEE Oran	Oran	03	Univ Oran 1	Oran

2.2.9 La jointure naturelle

Notée $R_a \bowtie R_b$, la jointure naturelle est appliquée à deux relations R_a et R_b. Elle donne une nouvelle relation dont les tuples sont les combinaisons des tuples de R_a et ceux de R_b qui satisfont la condition d'égalité entre les attributs de même nom. Le schéma de la nouvelle relation est la concaténation des schémas de R_a et de R_b, mais les attributs communs ne sont pas répétés.

Remarques :

1. Dans le cas de non-existence d'attributs communs entre R_a et R_b, l'application de la jointure naturelle est équivalente à l'application d'un produit cartésien entre ces deux relations.
2. Dans le cas d'existence d'attributs communs entre R_a et R_b, l'application de la jointure naturelle est équivalente à la combinaison d'un produit cartésien, d'une sélection et d'une projection.

Exemple : En appliquant la jointure naturelle $R_{16} = \text{Bachelier} \bowtie \text{Établissement}$, on obtient :

Relation R_{16}

numBach	nom	prénom	ville	moyenne	numEta	nomEta
03	Berrahou	Malika	Oran	12,00	02	ESGEE Oran
03	Berrahou	Malika	Oran	12,00	03	Univ Oran 1

On peut remarquer que la relation R_{16} est quasiment similaire à la relation R_{11}. La seule différence est la non répétition de l'attribut commun "ville".

Pour obtenir le nom et prénom des bacheliers qui ont postulé à un établissement qui se trouve dans leur ville, on pourrait appliquer :

$$R_{17} = \pi_{\text{nom,prénom}} \Big(\text{Bachelier} \bowtie \text{Postuler} \bowtie \text{Établissement}\Big)$$

Relation R_{17}

nom	prénom
Berrahou	Malika

2.2.10 La division

Notée $R_a \div R_b$, la division est appliquée à deux relations R_a et R_b, avec le schéma de R_b strictement inclus dans celui de R_a. Elle donne une nouvelle relation qui comporte les attributs appartenant à R_a mais n'appartenant pas à R_b et un ensemble des tuples qui concaténés à ceux de R_b donnent toujours un tuple de R_a.

Exemple : Pour obtenir les numéros des bacheliers qui ont postulé à tous les établissements, on appliquerait :

$$R_{18} = \pi_{\text{numBach,numEta}}\,(\text{Postuler}) \div \pi_{\text{numEta}}\,\Big(\text{Établissement}\Big)$$

Il en résulterait :

Relation R_{18}

numBach
01

2.3 Exercices

Exercice 14

1. Quelle est la condition sur deux relations R_1 et R_2 pour que l'opération $R_1 \cap R_2$ puisse être mise en place ?

2. Les deux requêtes $\pi_A\,(\sigma_{B='b'}(R))$ et $\sigma_{B='b'}\,(\pi_A(R))$ sont-elles équivalentes ?

3. Les deux requêtes $\sigma_{A=B}(R \times R')$ et $R \bowtie_{A=B} R'$ sont-elles équivalentes ?

4. L'effet de la jointure naturelle est équivalent à la combinaison de trois autres opérateurs. Quels sont-ils ?

Exercice 15 Soient les relations R_1 et R_2 :

R_1

A	B	C
a_1	b_1	c_1
a_3	b_1	c_2
a_2	b_3	c_1

R_2

A	B
a_1	b_1
a_1	b_3
a_3	b_1

1. Proposer des clés primaires pour les deux relations R_1 et R_2.

2. Pour chacune des requêtes ci-dessous, donner le nombre d'attributs de la nouvelle relation obtenue, ou préciser si la requête est fausse :

   a) $R_1 \cap R_2$

   b) $R_1 \times R_2$

   c) $R_1 \bowtie R_2$

d) $R_1 \div R_2$

3. Quels sont les résultats des requêtes suivantes :

a) $R_3 = \pi_{\mathrm{B}} \left(\sigma_{\mathrm{C}='c'_1}(R_1) \right)$

b) $R_4 = \pi_{\mathrm{C}} \left(\sigma_{(\mathrm{A}='a'_1) \vee (\mathrm{B}='b'_1)}(R_1) \right)$

c) $R_5 = \left(\pi_{\mathrm{A,B}} \left(\sigma_{\mathrm{C} \neq 'c'_1} (R_1) \right) \right) \cap R2$

Exercice 16

Soit la relation Etudiant avec le schéma (num, nom, prénom, ville) et qui contient 20 tuples. Les nouvelles relations obtenues par les requêtes suivantes auront combien de tuples au minimum et au maximum ?

1. $\pi_{\mathrm{num}}(\mathrm{Etudiant})$

2. $\pi_{\mathrm{nom}}(\mathrm{Etudiant})$

3. $\mathrm{Etudiant} \bowtie \mathrm{Etudiant}$

Exercice 17

En considérant les relations de la base de données utilisée pour le cours (celle de la gestion des nouveaux bacheliers, voir la page 40), expliquer ce que signifient les requêtes suivantes :

1. $\pi_{\mathrm{nomEta}}(\text{Établissement} \bowtie \sigma_{\text{intitulé='MI'}}(\text{Filière}))$

2. $\pi_{\mathrm{nom,prénom}}((\pi_{\mathrm{ville}}(\mathrm{Bachelier}) - \pi_{\mathrm{ville}}(\text{Établissement})) \bowtie \mathrm{Bachelier})$

3. $\pi_{\mathrm{nom,prénom}}((\pi_{\mathrm{numBach}}(\mathrm{Bachelier}) - \pi_{\mathrm{numBach}}(\mathrm{Postuler})) \bowtie \mathrm{Bachelier})$

Exercice 18

Soit le MLD composé des trois relations suivantes :

— personne(nom, age)
— mange(nom*, pizza)
— sert(pizzeria, pizza, prix)

Exprimer, en algèbre relationnelle, les requêtes permettant d'obtenir :

1. La liste de toutes les pizzas servies.

2. La liste de toutes les personnes qui ont plus de 20 ans.

3. La liste de toutes les pizzas mangées par au moins une personne qui a 20 ans.

4. La liste de toutes les personnes qui ont moins de 25 ans et qui ont mangé au moins une pizza servie par la pizzeria 111.

Exercice 19 [1]

Soit le MLD composé des deux relations suivantes :

— Film(titre, pays, année, réalisateur, durée)
— Jouer(titre, acteur)

Exprimer, en algèbre relationnelle, les requêtes permettant d'obtenir :

1. La liste des films algériens en affichant uniquement les attributs suivants : titre, année, réalisateur.

2. Les années de sortie des films où DE NIRO est acteur.

3. Les acteurs qui ont tourné avec RACHEDI comme réalisateur.

4. Tous les acteurs qui ont joué avec PORTMAN.

5. La liste des films où le réalisateur est aussi acteur.

6. Les réalisateurs qui ne jouent dans aucun de leur film.

7. Les acteurs qui jouent dans tous les films de TRUFFAUT.

Exercice 20

Soit le MLD composé des trois relations suivantes :

— Site(id_site, nom_site, adresse_site, ville_site, type_site)
— Guide(id_guide, nom_guide, prénom_guide, téléphone_guide, ville_guide)
— Visite(id_guide*,id_site*)

Exprimer, en algèbre relationnelle, les requêtes permettant d'obtenir :

1. La liste des sites touristiques de la ville de Tiaret.

2. Les numéros de téléphone des guides touristiques qui proposent des visites à Ghardaïa.

1. https ://stph.scenari-community.org/idl-bd/7/co/alg1e101.html

3. Les noms et prénoms des guides qui proposent au moins une visite dans une ville où ils n'habitent pas.

4. Les noms et prénoms des guides qui ne proposent aucune visite dans la ville où ils habitent.

Exercice 21

Soit la base de données d'une compagnie aérienne ayant le schéma (MLD) suivant :

— Avion(numAvion,numType*)
— Type(numType,nomType,capacité)
— Pilote(numPilote,nom,prénom,téléphone,salaire)
— Vol(numVol,numAvion*,numPilote*,villeDépart,villeArrivée,heureDépart,heureArrivée)

Exprimer, en algèbre relationnelle, les requêtes permettant d'obtenir :

1. Les noms et prénoms de tous les pilotes.

2. Les numéros des pilotes ayant effectué au moins un vol au départ de Djanet.

3. Les numéros des avions dont la capacité dépasse 400 places.

4. Les salaires des pilotes ayant déjà fait un vol d'un avion de capacité égale à 800 places.

5. Les noms et prénoms des pilotes qui ont effectué au moins un vol avec tous les avions de la compagnie.

2.4 Solutions des exercices

Solution de l'exercice 14

1. La condition sur deux relations R_1 et R_2 pour que l'opération $R_1 \cap R_2$ puisse être mise en place est que les deux relations aient le même schéma.

2. Non.

3. Oui.

4. L'effet de la jointure naturelle est équivalent à la combinaison des trois opérateurs : produit cartésien, sélection et projection.

Solution de l'exercice 15

1. Clé primaire pour la relation R_1 : A.
   Clé primaire pour la relation R_2 : A, B.

2. a) $R_1 \cap R_2$: requête fausse. Les deux relations n'ont pas le même schéma.

   b) $R_1 \times R_2$: requête valide. Relation résultante à 5 attributs.

   c) $R_1 \bowtie R_2$: requête valide. Relation résultante à 3 attributs.

   d) $R_1 \div R_2$: requête valide. Relation résultante à 1 attribut.

3. R_3

B
b_1
b_3

4. R_4

C
c_1
c_2

5. R_5

A	B
a_3	b_1

Solution de l'exercice 16

1. π_{num}Etudiant : cette relation contiendra exactement 20 tuples (au minimum 20 et au maximum 20).

2. π_{nom}Etudiant : cette relation contiendra au minimum 1 tuples (en considérant qu'une valeur pour l'attribut nom est obligatoire pour chaque tuple de la relation Etudiant) et au maximum 20 tuples.

3. Etudiant $\bowtie$ Etudiant : cette relation contiendra exactement 20 tuples (au minimum 20 et au maximum 20).

Solution de l'exercice 17

1. Noms des établissements qui proposent la filière MI.

2. Noms et prénoms des bacheliers qui habitent des villes où il n'existe aucun établissement.

3. Noms et prénoms des bacheliers qui n'ont postulé à aucun établissement.

Solution de l'exercice 18

1. $\pi_{\text{pizza}}(\text{sert})$
2. $\pi_{\text{nom}}(\sigma_{\text{age}>20}(\text{personne}))$
3. $\pi_{\text{pizza}}((\sigma_{\text{age}=20}(\text{personne}) \bowtie \text{mange})$
4. $\pi_{\text{nom}}(\sigma_{\text{age}<25}(\text{personne}) \bowtie \text{mange} \bowtie \sigma_{\text{pizzeria}='111'}(\text{sert}))$

Solution de l'exercice 19

1. $\pi_{\text{titre,année,réalisateur}}(\sigma_{\text{pays='Algérie'}}(\text{Film}))$
2. $\pi_{\text{année}}(\sigma_{\text{acteur='DE NIRO'}}(\text{Jouer}) \bowtie \text{Film})$
3. $\pi_{\text{acteur}}(\sigma_{\text{réalisateur='RACHEDI'}}(\text{Film}) \bowtie \text{Jouer})$
4. $\pi_{\text{acteur}}\left(\rho_{[\text{acteur :acteurP}]}\left(\sigma_{\text{acteur='PORTMAN'}}(\text{Jouer})\right) \bowtie \sigma_{\text{acteur}\neq\text{'PORTMAN'}}(\text{Jouer})\right)$
5. $\pi_{\text{titre}}\left(\sigma_{\text{acteur=réalisateur}}(\text{Jouer} \bowtie \text{Film})\right)$
6. $\pi_{\text{réalisateur}}(\text{Film}) - \pi_{\text{réalisateur}}(\text{Film} \bowtie_{\text{Film.titre=Jouer.titre}\wedge\text{réalisateur=acteur}} \text{Jouer})$
7. $\text{Jouer} \div \pi_{\text{titre}}(\sigma_{\text{réalisateur='TRUFFAUT'}}(\text{Film}))$

Solution de l'exercice 20

1. $\sigma_{\text{ville\_site='Tiaret'}}(\text{Site})$
2. $\pi_{\text{téléphone\_guide}}(\sigma_{\text{ville\_site='Ghardaia'}}(\text{Site}) \bowtie \text{Visite} \bowtie \text{Guide})$
3. $\pi_{\text{nom\_guide,prénom\_guide}}(\sigma_{\text{ville\_site}\neq\text{ville\_guide}}(\text{Site} \bowtie \text{Visite} \bowtie \text{Guide}))$
4. $\pi_{\text{nom\_guide,prénom\_guide}}((\pi_{\text{id\_guide}}(\text{Guide}) - \pi_{\text{id\_guide}}(\sigma_{\text{ville\_site=ville\_guide}}(\text{Site} \bowtie \text{Visite} \bowtie \text{Guide}))) \bowtie \text{Guide})$

Solution de l'exercice 21

1. $\pi_{\text{nom,prénom}}(\text{Pilote})$
2. $\pi_{\text{numPilote}}(\sigma_{\text{villeDépart='Djanet'}}(\text{Vol}))$
3. $\pi_{\text{numAvion}}(\text{Avion} \ltimes (\sigma_{\text{capacité}>400}(\text{Type})))$
4. $\pi_{\text{salaire}}(\text{Pilote} \ltimes \text{Vol} \ltimes \text{Avion} \ltimes (\sigma_{\text{capacité}=800}(\text{Type})))$

5. $\pi_{\text{nom,prénom}}\left(\left(\left(\pi_{\text{numPilote,numAvion}}(\text{Vol})\right) \div \left(\pi_{\text{numAvion}}(\text{Avion})\right)\right) \bowtie (\text{Pilote})\right)$

CHAPITRE 3

LE LANGAGE SQL

3.1 Introduction

Une fois la modélisation du réel faite selon les besoins de l'utilisateur, et les relations prêtes, nous pouvons passer à la création de notre base de données puis à sa manipulation. Pour cela, nous devons utiliser un système de gestion de base de données (SGBD).

Un SGBD est un logiciel qui permet de créer une base de données et de la contrôler. Il est basé sur un langage informatique qui exprime les requêtes/ordres de l'utilisateur/développeur/administrateur (pour plus de détails, revenir au chapitre 0).

Pour notre cours et son TP, nous considérerons le SGBD relationnel MySQL qui utilise le langage SQL.

Le système de gestion de bases de données relationnelles (SGBDR) MySQL domine le marché des bases de données open-source, et est fortement utilisé pour la création de site web/applications à gros volumes de données tel que Facebook, Wikipedia ou YouTube [6]. Une version gratuite de MySQL est disponible pour toute utilisation personnelle ou de recherche.

SQL, pour **S**tructured **Q**uery **L**angage, est quant à lui un langage de requêtes utilisé pour interagir avec des bases de données relationnelles et il est supporté par un grand nombre de SGBDRs (MySQL comme précisé précédemment, Microsoft SQL Server, Oracle, Microsoft Access, etc.). Chaque SGBD possède sa propre version de SQL, mais celles-ci sont assez similaires [6].

SQL est grandement basé sur le modèle relationnel théorique, mais il présente toutefois quelques "déviations". Exemple : SQL permet la possibilité d'avoir des doublons dans une table.

SQL peut être décomposé en :

— une partie "*Langage de définition de données* " (LDD) qui intervient au niveau du schéma d'une BDD. Le LDD permet ainsi de créer, de modifier et de supprimer des BDDs ou des

tables ;

— une partie "*Langage de manipulation de données* " (LMD) qui intervient au niveau des données elles-mêmes. Le LMD permet alors d'insérer, de mettre à jour et de supprimer des données. Il permet aussi et surtout d'interroger les données ;

— une partie "*Langage de contrôle de transaction* " (LCT) qui permet de commencer et de terminer des transactions (voir la section Terminologie) ;

— une partie "*Langage de contrôle des données* " (LCD) qui permet d'autoriser ou d'interdire l'accès à certaines données à certaines personnes.

Dans notre cours, nous nous intéresserons uniquement aux deux parties : "Langage de définition de données" (LDD) et "Langage de manipulation de données" (LMD).

3.2 Signification des symboles

Signification des symboles utilisés dans le chapitre lors de la présentation des syntaxes des requêtes :

— `BLABLA` : signifie que "blabla" est un mot clé du langage SQL ;

— `[blabla]` : signifie que "blabla" est optionnel (c.à.d. qu'il peut être utilisé ou non dans la requête). Les crochets ([]) ne devront pas apparaitre dans la requête.

— `<blabla>` : signifie que "blabla" est à remplacer par un terme indiqué dans la requête. Il pourra s'agir d'un nom de BDD, d'un nom d'attribut, d'une valeur numérique, d'une constante de type chaîne de caractère à insérer ou bien à comparer, etc. Les chevrons (<>) ne devront pas apparaitre dans la requête.

Les exemples données dans ce chapitre se baseront sur la même base de données que le chapitre 2.

3.3 Terminologie

Les notions suivantes sont utilisées au fur et à mesure dans le chapitre. Vous pourrez revenir à leur définition plus tard.

Requête : ordre donné à la BDD et ayant comme résultat un changement d'état de la BDD ou un affichage de résultats.

Transaction : suite de requêtes qui font passer la base de données d'un état à un autre. Un exemple de transaction sur une BDD bancaire est un transfert d'argent d'un compte à un autre, où il y a notamment une requête de retrait d'argent sur le premier compte et une autre requête de dépôt sur le second.

Clause : partie d'une requête.

Contrainte : règle qui permet de limiter le type de données à insérer dans une table.

3.4 Langage de définition

Les requêtes de définition de données de SQL sont les suivantes :

— **CREATE** : permet la création de bases de données ou de tables,

— **ALTER** : permet la modification de bases de données ou de tables,

— **DROP** : permet la suppression de bases de données ou de tables.

3.4.1 Création d'une base de données

La syntaxe de création d'une base de données est la suivante :

```
CREATE DATABASE <nom_bdd>;
```

avec :

→ `nom_bdd` : Nom de la base de données à créer.

Cette requête permet de créer la base de données "nom_bdd" si aucune base de données avec le même nom n'existe encore.

Remarques :

1. SQL est sensible à la casse en ce qui concerne les noms des bases de données, des tables et des attributs, mais ne l'est pas en ce qui concerne ses mots clés. (utilisation des majuscules/minuscules)
2. L'accès à la base de données "nom_bdd" se fait à travers la requête :
   ```
   USE <nom_bdd>;
   ```
3. Il est possible de consulter la liste des bases de données disponibles à travers la requête :
   ```
   SHOW DATABASES;
   ```

Exemple : Afin de créer la base données nommée "bac", la requête est :

```
CREATE DATABASE bac;
```

Avant de pouvoir travailler dessus, il est nécessaire de la sélectionner à travers :

```
USE bac;
```

Ainsi, la requête **CREATE** `DATABASE bac` sera exécutée une seule fois. Par contre, la requête `USE bac` devra être exécutée à chaque fois qu'on voudra travailler dessus.

3.4.2 Suppression d'une base de données

La syntaxe de suppression d'une base de données est la suivante :

```
DROP DATABASE <nom_bdd>;
```

Cette requête permet de supprimer la base de données `nom_bdd` si une telle base de données existe.

3.4.3 Création d'une table

Types de données

Parmi les types de données disponibles en SQL, nous considérons les types suivants :

— **`INT`** : pour les entiers,

— **`FLOAT`** : pour les flottants,

— **`CHAR(<n>)`** : pour les chaînes ayant une taille prédéterminée fixe de `n` caractères, avec `n` ne dépassant pas la valeur 255,

— **`VARCHAR(<n>)`** : pour les chaînes ayant une taille variable, mais un maximum de `n` caractères, avec `n` ne dépassant pas la valeur 65535,

— **`DATE`** : pour les dates,

— **`TIME`** : pour l'heure.

Création d'une table

La syntaxe de création d'une table est la suivante :

```
CREATE TABLE <nom_table>(
  <att1> <type_att_1>,
  <att2> <type_att_2>,
  ...
  <attn> <type_att_n>,
  [PRIMARY KEY(<cle_primaire>),]
  [FOREIGN KEY(<cle_etrangere_1>)
   REFERENCES <nom_autre_table_1>(<att_autre_table_1>),]
  [FOREIGN KEY(<cle_etrangere_2>)
   REFERENCES <nom_autre_table_2>(<att_autre_table_2>),]
    ...
  [FOREIGN KEY(<cle_etrangere_m>)
   REFERENCES <nom_autre_table_m>(<att_autre_table_m>)]
);
```

avec :

→ `nom_table` : Nom de la table à créer,

→ `att_i` : Attribut numéro `i` de la table,

→ `type_att_i` : Type de l'attribut numéro `i`,

→ `cle_primaire` : Clé primaire de la table. Elle peut être composée d'un seul attribut ou d'une liste d'attributs séparés par des virgules,

→ `cle_etrangere_i` : Clé étrangère numéro `i`,

→ `nom_autre_table_i` : Table d'origine de la clé étrangère numéro `i`,

→ `att_autre_table_i` : Nom de l'attribut dans la table d'origine de la clé étrangère numéro `i`.

Remarques :

1. Une fois une table crée, son schéma peut être consulté en utilisant la requête :

```
DESCRIBE <nom_table>;
```

2. Il est possible de consulter la liste de tables de la base de données actuelle à travers la requête :

```
SHOW TABLES;
```

3. L'ajout d'une clé étrangère à une table est considérée comme une contrainte ayant un nom. Il est possible de consulter la liste des noms de toutes les contraintes d'une table à travers la requête :

```
SHOW CREATE TABLE <nom_table>;
```

Exemple : Les tables de la base de données "bac" dont les schémas sont :

— Etablissement(<u>numEta</u>, nomEta, ville)

— Bachelier(<u>numBach</u>, nom, prenom, ville, moyenne)

— Filière(<u>numFilière</u>, intitulé)

— Postuler(<u>numEta*, numBach*, numFiliere*</u>)

peuvent être crées dans la base de données `bac` à travers les requêtes :

```
CREATE TABLE Etablissement(
  numEta INT,
  nomEta VARCHAR(20),
  ville VARCHAR(10),
  PRIMARY KEY(numEta)
);
CREATE TABLE Bachelier(
  numBach INT,
  nom VARCHAR(20),
  prenom VARCHAR(20),
  ville VARCHAR(10),
  moyenne FLOAT,
  PRIMARY KEY(numBach)
);
CREATE TABLE Filiere(
  numFiliere INT,
  intitule VARCHAR(20),
  PRIMARY KEY(numFiliere)
);
CREATE TABLE Postuler(
```

```
  numBach INT,
  numEta INT,
  numFiliere INT,
  PRIMARY KEY(numEta,numBach,numFiliere),
  FOREIGN KEY(numBach) REFERENCES Bachelier(numBach),
  FOREIGN KEY(numEta) REFERENCES Etablissement(numEta),
  FOREIGN KEY(numFiliere) REFERENCES Filiere(numFiliere)
);
```

3.4.4 Modification d'une table

La modification d'une table se fait à travers la clause **ALTER** à laquelle on associera d'autres clauses.

Modification du nom d'une table

La modification du nom d'une table se fait en associant la clause `RENAME TO` à la clause **ALTER**, ce qui donne :

```
ALTER TABLE <ancien_nom>
RENAME TO <nouveau_nom>;
```

Modification du nom d'un attribut

La modification du nom d'un attribut d'une table se fait en associant la clause `CHANGE` à la clause **ALTER**, ce qui donne :

```
ALTER TABLE <nom_table>
CHANGE <ancien_nom> <nouveau_nom> <type>;
```

Remarque Le type de l'attribut renommé doit être précisé et rester inchangé.

Modification du type d'un attribut

La modification du type d'un attribut d'une table se fait en associant la clause `MODIFY COLUMN` à la clause `ALTER`, ce qui donne :

```
ALTER TABLE <nom_table>
MODIFY COLUMN <att> <nouveau_type>;
```

Ajout d'un attribut ou d'une clé à une table

L'ajout d'un nouvel attribut ou d'une nouvelle clé se fait en associant la clause **ADD** à la clause **ALTER**. Ainsi :

— L'ajout d'un attribut à une table se fait à travers la requête :

```
ALTER TABLE <nom_table>
ADD <nouvel_att> <type_nouvel_att>;
```

— L'ajout d'une clé primaire à une table se fait à travers la requête :

```
ALTER TABLE <nom_table>
ADD PRIMARY KEY(<cle_primaire>);
```

— L'ajout d'une clé étrangère à une table se fait à travers la requête :

```
ALTER TABLE <nom_table>
ADD FOREIGN KEY(<cle_etrangere>)
REFERENCES <nom_autre_table>(<att_autre_table>);
```

Exemple L'ajout de l'attribut `dateNaissance` de type **DATE** à la table `Bachelier`, peut se faire comme suit :

```
ALTER TABLE Bachelier
ADD dateNaissance DATE;
```

Suppression d'un attribut ou d'une clé d'une table

La suppression d'un attribut d'une table se fait en associant la clause **DROP** à la clause **ALTER**. Ainsi :

— La suppression d'un attribut d'une table se fait à travers la requête :

```
ALTER TABLE <nom_table>
DROP COLUMN <att>;
```

— La suppression d'une clé primaire d'une table se fait à travers la requête :

```
ALTER TABLE <nom_table>
DROP PRIMARY KEY;
```

— La suppression d'une clé étrangère d'une table se fait à travers la requête :

```
ALTER TABLE <nom_table>
DROP FOREIGN KEY <nom_contrainte>;
```

Suppression d'une table

La suppression d'une table se fait à travers la requête :

```
DROP TABLE <nom_table>;
```

3.5 Langage de manipulation

Les requêtes de manipulation de données sont les suivantes :

— **INSERT** : permet l'insertion de données,
— **UPDATE** : permet la mise à jour de données,
— **DELETE** : permet la suppression de données,
— **SELECT** : permet l'interrogation de données.

3.5.1 Insertion des données

L'insertion des données, c.à.d. de tuples, dans une table se fait à travers la requête :

```
INSERT INTO <nom_table>
VALUES <tuple_1>,
     <tuple_2>,
      ...
     <tuple_n>;
```

avec :

→ `tuple_i` : tuple à insérer. Il doit être écrit sous la forme `(val_att_1, val_att_2,...val_att_m)`, en respectant l'ordre des `m` attributs du schéma de la table concernée.

Remarques

1. Le séparateur des valeurs de type **FLOAT** est le point et non la virgule, ex. 15.35 et non 15,35,
2. Les valeurs de type **CHAR** et **VARCHAR** doivent être manipulées en les mettant entre quottes, ex. `'chaine'`,
3. Les valeurs de type **DATE** doivent également être manipulées en les mettant entre quottes. De plus, leur format est `'aaaa-mm-jj'`,
4. Les valeurs de type **TIME** doivent également être manipulées en les mettant entre quottes. De plus, leur format est `'hh:mm:ss'`,
5. Il est possible d'utiliser le mot clé **NULL** afin de laisser la valeur d'un attribut vide.

Exemple L'insertion du premier tuple dans la table `Etablissement` peut se faire à travers la requête :

```
INSERT INTO Etablissement
VALUES (01, 'Univ Alger 1', 'Alger');
```

L'insertion des deux autres tuples dans la même table peut se faire à travers une seule requête :

```
INSERT INTO Etablissement
VALUES (02, 'ESGEE Oran', 'Oran'),
     (03, 'Univ Oran 1', 'Oran');
```

L'insertion d'un tuple dans la table `Bachelier` peut se faire à travers la requête :

```
INSERT INTO Bachelier
VALUES (01, 'Belaid', 'Houda', 'Mascara', 15.35, '2003-12-03');
```

L'insertion du deuxième tuple dans la table `Bachelier` sans préciser la valeur de l'attribut `dateNaissance` peut se faire à travers la requête :

```
INSERT INTO Bachelier
VALUES (02, 'Ammar', 'Ali', 'Meliana', 10.01, NULL);
```

3.5.2 Mise à jour de données

La mise à jour de données existantes dans une table se fait à travers la requête :

```
UPDATE <nom_table>
SET <att_1>=<val_1>,
  <att_2>=<val_2>,
  ...
  <att_n>=<val_n>
[WHERE <condition>];
```

avec :

→ `att_i` : attribut à modifier,

→ `val_i` : nouvelle valeur pour l'attribut `att_i`,

→ `condition` : à préciser si la modification doit se faire uniquement sur les tuples qui satisfont cette condition.

Remarques

1. Parmi les opérateurs binaires de comparaisons classiques qu'il est possible d'utiliser dans la condition de la clause **`WHERE`**, nous avons : `<` (inférieur à), `>` (supérieur à), `<=` (inférieur ou égal à), `>=` (supérieur ou égal à), `=` (égal à), `!=` (différent de).
2. L'opérateur binaire **`LIKE`** permet de comparer des chaînes de caractères à des schémas. Dans ces schémas :
   — le caractère `%` peut remplacer n'importe quelle chaîne,
   — le caractère `_` peut remplacer n'importe quel caractère.
3. Les opérateurs binaires **`IN`** et **`NOT IN`** permettent de vérifier l'appartenance ou non d'une valeur à un ensemble de valeurs.
4. Les opérateurs unaires `IS` **`NULL`** et `IS` **`NOT NULL`** permettent de vérifier si une attribut est à **`NULL`** ou non.
5. Plusieurs conditions peuvent être combinées à l'aide du **`AND`** (et logique) et du **`OR`** (ou logique).

Exemples Pour la suite des exemples, nous considérerons que la base de données est dans l'état suivant et que chaque modification ou interrogation est faite sur cette version de la base de données.

Table Bachelier

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
02	Ammar	Ali	Meliana	10.01	2003-05-25
03	Berrahou	Malika	Oran	12.00	NULL
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29

Table Établissement

numEta	nomEta	ville
01	Univ Alger 1	Alger
02	ESGEE Oran	Oran
03	Univ Oran 1	Oran

Table Filière

numFilière	intitulé
01	Architecure
02	MI
03	Prépa ST
04	ST
05	Droit

Table Postuler

numBach	numEta	numFilière
01	01	01
01	02	03
01	03	04
02	01	05
03	01	05
03	03	02

Si on souhaite mettre à **NULL** les dates de naissance de tous les bacheliers, on peut appliquer la requête :

```
UPDATE Bachelier
SET dateNaissance=NULL;
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	NULL
02	Ammar	Ali	Meliana	10.01	NULL
03	Berrahou	Malika	Oran	12.00	NULL
04	Maaziz	Houda	Tlemcen	11.58	NULL

Si on souhaite modifier la ville du bachelier n°4 en lui affectant la valeur 'Blida' :

```
UPDATE Bachelier
SET ville = 'Blida'
WHERE numBach = 4;
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
02	Ammar	Ali	Meliana	10.01	2003-05-25
03	Berrahou	Malika	Oran	12.00	NULL
04	Maaziz	Houda	Blida	11.58	2004-02-29

Si on souhaite supprimer le prénom de tous les bacheliers qui n'habitent pas à 'Meliana' :

```
UPDATE Bachelier
SET prenom = NULL
WHERE ville != 'Meliana';
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	NULL	Mascara	15.35	2000-02-15
02	Ammar	Ali	Meliana	10.01	2003-05-25
03	Berrahou	NULL	Oran	12.00	NULL
04	Maaziz	NULL	Tlemcen	11.58	2004-02-29

Si on souhaite transformer le prénom en 'Mohamed' pour tous les bacheliers dont le nom commence par 'B' :

```
UPDATE Bachelier
SET prenom = 'Mohamed'
WHERE nom LIKE 'B%' ;
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Mohamed	Mascara	15.35	2000-02-15
02	Ammar	Ali	Meliana	10.01	2003-05-25
03	Berrahou	Mohamed	Oran	12.00	NULL
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29

Si on souhaite mettre à NULL les prénoms qui sont de longueur égale à 3 caractères :

```
UPDATE Bachelier
SET prenom = NULL
WHERE prenom LIKE '___' ;
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
02	Ammar	NULL	Meliana	10.01	2003-05-25
03	Berrahou	Malika	Oran	12.00	NULL
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29

Si on souhaite mettre à jour le prénom à la valeur 'Malika' pour les bacheliers dont le numéro appartient à l'ensemble {1,2} :

```
UPDATE Bachelier
SET prenom = 'Malika'
WHERE numBach IN (1,2) ;
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Malika	Mascara	15.35	2000-02-15
02	Ammar	Malika	Meliana	10.01	2003-05-25
03	Berrahou	Malika	Oran	12.00	NULL
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29

Si on souhaite mettre à jour le prénom à la valeur 'Malika' pour les bacheliers dont le numéro n'appartient pas à l'ensemble {1,2} :

```
UPDATE Bachelier
SET prenom = 'Malika'
WHERE numBach NOT IN (1,2) ;
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
02	Ammar	Ali	Meliana	10.01	2003-05-25
03	Berrahou	Malika	Oran	12.00	NULL
04	Maaziz	Malika	Tlemcen	11.58	2004-02-29

Si on souhaite mettre à NULL les prénoms des bacheliers dont la date de naissance n'est pas précisée :

```
UPDATE Bachelier
SET prenom = NULL
WHERE dateNaissance IS NULL ;
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
02	Ammar	Ali	Meliana	10.01	2003-05-25
03	Berrahou	NULL	Oran	12.00	NULL
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29

Si on souhaite mettre à jour la date de naissance et la ville du bachelier numéro 2 dans le cas où sa ville est **NULL** ou que son prénom contient la chaîne `'li'`, on peut appliquer :

```
UPDATE Bachelier
SET dateNaissance='2002-12-31',
  ville='Bejaia'
WHERE (numBach=2) AND ((ville IS NULL) OR (prenom LIKE '%li%'));
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
02	Ammar	Ali	Bejaia	10.01	2002-12-31
03	Berrahou	Malika	Oran	12.00	NULL
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29

3.5.3 Suppression de données

La suppression de données, c.à.d. de tuples, se fait à travers la requête :

```
DELETE FROM <nom_table>
[WHERE <condition>];
```

avec :

→ `condition` : à préciser si la suppression doit se faire uniquement sur les tuples qui satisfont cette condition.

Exemples Si on souhaite supprimer tous les bacheliers d'Oran :

```
DELETE FROM bachelier
WHERE ville='Oran';
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
02	Ammar	Ali	Meliana	10.01	2003-05-25
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29

Si on souhaite supprimer tous les bacheliers mais qu'on souhaite tout de même garder la table `Bachelier` :

```
DELETE FROM Bachelier;
```

Table Bachelier mise à jour

numBach	nom	prénom	ville	moyenne	dateNaissance

3.5.4 Interrogation des données

L'interrogation de données se fait à travers la clause **`SELECT`**. Elle permet d'obtenir une nouvelle table. Une requête d'interrogation de données en SQL repose sur les principes de base des opérateurs de l'algèbre relationnelle.

Interrogation d'une table

L'application du **SELECT** sur une table se fait de la manière suivante :

```
SELECT [DISTINCT] {<att_1>, <att_2>,..., <att_n>|*}
FROM <table>
[WHERE <condition>]
[ORDER BY <att> {ASC|DESC}];
```

avec :

→ la clause **SELECT** : permet de préciser la liste d'attributs qui formeront le schéma de la nouvelle table : `<att_1>, <att_2>,..., <att_n>`. Ils appartiennent à la table originale ou ils sont calculés. Le caractère spécial `*` peut être utilisé au lieu de la liste d'attributs lorsqu'on souhaite garder tous les attributs de la table originale. L'option **DISTINCT** permet d'éliminer les doublons,

→ la clause **FROM** : permet de préciser la table originale sur laquelle la requête est appliquée : `<table>`,

→ la clause optionnel **WHERE** : permet de préciser une condition que devront respecter les tuples obtenus dans la nouvelle table,

→ la clause optionnel **ORDER** : permet d'ordonner les tuples obtenus par ordre croissant (**ASC**) ou décroissant (**DESC**) en fonction de l'attribut `att`.

Remarques

1. Sans la précision de la clause **ORDER BY**, l'ordre des tuples obtenu est aléatoire et peut différer d'une exécution à une autre.
2. Plusieurs attributs peuvent être spécifiés dans la clause **ORDER BY**.
3. Il est possible de renommer les attributs ou les tables en utilisant le mot clé **AS**.
4. Le résultat d'une requête **SELECT** est une nouvelle table. De ce fait, la requête **SELECT** peut être elle même imbriquée dans une autre requête en tant que table.

Exemples Afin d'afficher tout le contenu de la table `Bachelier`, on peut appliquer la requête :

```
SELECT *
FROM Bachelier;
```

Résultat de la requête d'interrogation sans précision d'attributs

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
02	Ammar	Ali	Meliana	10.01	2003-05-25
03	Berrahou	Malika	Oran	12.00	NULL
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29

Afin obtenir uniquement les prénoms de tous les bacheliers en gardant les doublons, la requête est :

```
SELECT prenom
FROM Bachelier;
```

Résultat de la requête d'interrogation avec précision d'attributs

prénom
Houda
Ali
Malika
Houda

Afin obtenir uniquement les prénoms de tous les bacheliers en supprimant les doublons, la requête est :

```
SELECT DISTINCT prenom
FROM Bachelier;
```

Résultat de la requête d'interrogation avec élimination des doublons

prénom
Houda
Ali
Malika

Afin d'obtenir les bacheliers dont la moyenne est supérieure ou égale à 12, on peut appliquer la requête :

```
SELECT *
FROM Bachelier
WHERE moyenne >= 12;
```

Résultat de la requête d'interrogation avec condition

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
03	Berrahou	Malika	Oran	12.00	NULL

Afin d'obtenir les bachelier dont le prénom se termine par "da", on peut appliquer la requête :

```
SELECT *
FROM Bachelier
WHERE prenom LIKE '%da';
```

Résultat de la requête d'interrogation avec condition

numBach	nom	prénom	ville	moyenne	dateNaissance
01	Belaid	Houda	Mascara	15.35	2000-02-15
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29

Afin d'obtenir les bacheliers dont la moyenne est supérieure à 10, ordonnés selon leur prénom par ordre croissant puis selon leur nom par ordre décroissant, on peut appliquer la requête :

```
SELECT *
FROM Bachelier
WHERE moyenne > 10
ORDER BY prenom ASC, nom DESC;
```

Résultat de la requête d'interrogation avec condition et ordre appliqué

numBach	nom	prénom	ville	moyenne	dateNaissance
02	Ammar	Ali	Meliana	10.01	2003-05-25
04	Maaziz	Houda	Tlemcen	11.58	2004-02-29
01	Belaid	Houda	Mascara	15.35	2000-02-15
03	Berrahou	Malika	Oran	12.00	NULL

Afin d'obtenir le nom, prénom et la moyenne des bacheliers sur 10 au lieu de 20, on peut appliquer la requête :

```
SELECT nom, prenom, moyenne/2
FROM Bachelier
```

Résultat de la requête d'interrogation avec opération arithmétique

nom	prénom	moyenne/2
Belaid	Houda	7.675
Ammar	Ali	5.005
Berrahou	Malika	6
Maaziz	Houda	5.79

Nous obtenons alors une nouvelle table avec le schéma (nom, prenom, moyenne/2). Afin de renommer le dernier attribut obtenu ; il est possible d'utiliser le mot clé **AS** comme suit :

```
SELECT nom, prenom, (moyenne/2) AS 'moy'
FROM Bachelier
```

Le schéma de la nouvelle table sera alors : (nom, prenom, moy).

Résultat de la requête d'interrogation avec renommage d'attribut

nom	prénom	moy
Belaid	Houda	7.675
Ammar	Ali	5.005
Berrahou	Malika	6
Maaziz	Houda	5.79

Produit cartésien

Un produit cartésien, ou une séries de produits cartésiens, peut être associé au **SELECT** en remplaçant **<table>** par une liste de tables séparées par des virgules dans la clause **FROM**,

comme suit :

```
SELECT [DISTINCT] {<att_1>, <att_2>,..., <att_n>|*}
FROM <table_1>, <table_2>, ... <table_m>
[WHERE <condition>]
[ORDER BY <att> {ASC|DESC}];
```

Remarque Un attribut commun `att` entre deux tables `table_i` et `table_j` devra être désigné respectivement par `table_i.att` et `table_j.att` dans la requête.

Exemple Afin d'obtenir le numéro, le nom et le prénom de tous les bacheliers et le numéro des filières auxquelles chacun a postulé, on peut appliquer :

```
SELECT Bachelier.numBach,nom, prenom, numFiliere
FROM Bachelier, Postuler
WHERE Bachelier.numBach = Postuler.numBach;
```

Résultat de la requête d'interrogation avec produit cartésien

Bachelier.numBach	nom	prénom	numFilière
01	Belaid	Houda	01
01	Belaid	Houda	03
01	Belaid	Houda	04
02	Ammar	Ali	05
03	Berrahou	Malika	05
03	Berrahou	Malika	02

La bachelière n°4 ne s'affiche pas dans le résultat puisqu'elle n'a postulé à aucune filière.

Jointure

La jointure peut également être associée au **SELECT** comme suit :

```
SELECT [DISTINCT] {<att_1>, <att_2>,..., <att_n>|*}
FROM <table_1> JOIN <table_2> ON <condition_1>
[WHERE <condition_2>]
[ORDER BY <att> {ASC|DESC}];
```

Exemples L'exemple précédant aurait pu être formulé en appliquant la jointure comme suit :

```
SELECT Bachelier.numBach, nom, prenom, numFiliere
FROM Bachelier JOIN Postuler ON Bachelier.numBach=Postuler.numBach;
```

Résultat de la requête d'interrogation avec jointure

Bachelier.numBach	nom	prénom	numFilière
01	Belaid	Houda	01
01	Belaid	Houda	03
01	Belaid	Houda	04
02	Ammar	Ali	05
03	Berrahou	Malika	05
03	Berrahou	Malika	02

Afin d'obtenir le nom et prénom des bacheliers avec une moyenne égale à 12 et le numéro des filières auxquelles chacun a postulé, on peut appliquer :

```
SELECT nom, prenom, numFiliere
FROM Bachelier JOIN Postuler ON Bachelier.numBach=Postuler.numBach
WHERE moyenne=12;
```

Résultat de la requête d'interrogation avec jointure et condition

nom	prénom	numFilière
Berrahou	Malika	05
Berrahou	Malika	02

Jointure naturelle

La jointure naturelle peut elle aussi être associée au **SELECT** comme suit :

```
SELECT [DISTINCT] {<att_1>, <att_2>,..., <att_n>|*}
FROM <table_1> NATURAL JOIN <table_2>
[WHERE <condition>]
[ORDER BY <att> {ASC|DESC}];
```

Exemple Le dernier exemple aurait pu être formulé en appliquant la jointure naturelle comme suit :

```
SELECT nom, prenom, numFiliere
FROM Bachelier NATURAL JOIN Postuler
WHERE moyenne=12;
```

Résultat de la requête d'interrogation avec jointure naturelle et condition

nom	prénom	numFilière
Berrahou	Malika	05
Berrahou	Malika	02

Afin d'obtenir le nom et le prénom des bacheliers avec une moyenne égale à 12 et l'intitulé des filières auxquelles chacun a postulé, on peut appliquer :

```
SELECT nom, prenom, intitule
FROM Bachelier NATURAL JOIN Postuler NATURAL JOIN Filiere
WHERE moyenne=12;
```

Résultat de la requête d'interrogation avec deux jointures naturelles

nom	prénom	intitulé
Berrahou	Malika	Droit
Berrahou	Malika	MI

Division

L'opérateur de division étudié en algèbre relationnelle n'a pas d'équivalent direct en SQL. Il est toutefois possible d'obtenir son résultat en utilisant la double négation.

Exemple Afin d'obtenir les numéros des bacheliers qui ont postulé à toutes les universités, il serait possible de procéder comme suit :

— Obtenir les numéros des bacheliers qui n'ont pas postulé à tous les établissements ;

— Obtenir les numéros des bacheliers qui n'apparaissent pas dans l'étape précédente.

La requête serait alors :

```
SELECT numBach
FROM Bachelier
WHERE numBach NOT IN(
  SELECT numBach
  FROM Bachelier, Etablissement
  WHERE (numBach, Etablissement.numEta) NOT IN (
    SELECT numBach, numEta
    FROM Postuler
  )
);
```

Fonctions d'agrégation

Les fonctions d'agrégation sont des clauses à associer au **SELECT** afin d'obtenir des résultats d'opérations statistiques sur un ensemble de tuples. Les principales fonctions sont :

COUNT() : retourne le nombre de tuples dans une table ou de valeurs non nulles dans une colonne ;

MAX() : retourne la valeur maximum d'une colonne ;

MIN() : retourne la valeur minimum d'une colonne ;

SUM() : retourne la somme sur un ensemble de tuples ;

AVG() : retourne la moyenne sur un ensemble de tuples ;

L'utilisation des fonctions d'agrégation se fait de la manière suivante :

```
SELECT <fonction>({<att>|*})
FROM <table>
[WHERE <condition>]
[ORDER BY <att> {ASC|DESC}];
```

en remplaçant "fonction" par la fonction d'agrégation correspondante.

Remarque L'utilisation de `COUNT(*)` retourne le nombre de tuples de la table.

Exemple Afin d'obtenir le nombre d'établissements dans la BDD, la requête à appliquer est :

```
SELECT COUNT(*)
FROM Etablissement ;
```

Résultat de la requête d'interrogation avec utilisation de la fonction d'agrégation de comptage

COUNT(*)
3

Afin d'obtenir le nombre d'établissements dans la BDD qui se trouvent à Oran, la requête à appliquer est :

```
SELECT COUNT(*)
FROM Etablissement
WHERE ville='Oran';
```

Résultat de la requête d'interrogation avec utilisation de la fonction d'agrégation COUNT et condition

COUNT(*)
2

Afin d'obtenir la meilleure moyenne entre les bacheliers, la requête à appliquer est :

```
SELECT MAX(moyenne)
FROM Bachelier;
```

Résultat de la requête d'interrogation avec utilisation de la fonction d'agrégation MAX

MAX(moyenne)
15.35

Afin d'obtenir la meilleure moyenne entre les bacheliers ayant postulé à au moins un établissement à Alger, la requête à appliquer est :

```
SELECT MAX(moyenne)
FROM (Bachelier NATURAL JOIN Postuler)
  JOIN Etablissement ON Postuler.numEta=Etablissement.numEta
WHERE Etablissement.ville='Alger';
```

Résultat de la requête d'interrogation avec utilisation de la fonction d'agrégation MAX, ainsi qu'une jointure, une jointure naturelle et une condition

MAX(moyenne)
15.35

Remarque La clause **GROUP BY** permet de regrouper les tuples selon la valeur d'une ou de plusieurs colonnes, puis d'appliquer les fonctions d'agrégation sur chaque groupe. La syntaxe devient alors :

```
SELECT <att_1>, <att_2>,..., <att_n>, <fonction>({<att>|*})
FROM <table>
[WHERE <condition>]
[GROUP BY <att_1'>, <att_2'>,..., <att_n'>]
[ORDER BY <att'> {ASC|DESC}];
```

Exemple Afin d'afficher la moyenne des moyennes des bacheliers pour chaque ville, et avec les noms des villes ordonnées alphabétiquement, la requête serait :

```
SELECT ville, AVG(moyenne)
FROM Bachelier
GROUP BY ville
ORDER BY ville ASC;
```

Résultat de la requête d'interrogation avec utilisation de la fonction d'agrégation AVG, GROUP BY et ORDER BY

ville	AVG(moyenne)
Blida	14.5
Mascara	15.35
Meliana	10.01
Oran	11
Tlemcen	11.58

Ainsi, les groupes formés par cette requête seront des groupes où la valeur de la colonne "ville" est identique. Pour chaque groupe, donc pour chaque ville, la moyenne des moyennes est affichée.

3.6 Exercices

Les exercices proposés dans cette section ont pour but d'être résolus par la pratique sur un appareil disposant d'un SGBDR MySQL qui est basé sur le langage SQL et de phpMySQL comme interface graphique.

Toutefois, ces exercices peuvent bien évidemment être résolus à l'aide d'une feuille et d'un stylo.

Dans le premier cas, pour certains exercices, des requêtes sont données au préalable et doivent être copiées/collées puis exécutées avant de donner les requêtes demandées par l'exercice. Les requêtes données peuvent être trouvées à la section 3.9.

Exercice 22

Soit la base de données avec le MLD suivant :

— Client(num_client, nom, prenom, ville)

— Compte(num_compte, num_client*, type)

— Operation(num_op, num_compte*, montant, informations)

En utilisant les requêtes du langage SQL :

1. Créer la base de données `'banque'`.
2. Après avoir accéder à votre nouvelle base de données, créer les 3 tables.
3. Peut-on supprimer la table `'Client'` ? Pourquoi ?
4. Supprimer la table `'Operation'`.

Exercice 23

Nous utiliserons pour cet exercice les requêtes de création des tables d'une base de données pour la gestion d'un festival du théâtre données à la section 3.9.1.

1. Consulter les requêtes données à la section 3.9.1.
2. Revenir à phpMyAdmin et créer la base de données `'festival'` puis y accéder.
3. Dans l'onglet SQL, faire un copier/coller ou bien saisir les requêtes données à la section 3.9.1 afin de créer les tables de la base de données.
4. En utilisant les requêtes du langage SQL, apporter les modifications suivantes à la base de données :
   (a) Ajouter l'attribut `'heure_debut'` à la table `'Programmer'`.
   (b) Modifier le type de l'attribut `'titre'` de la table `'Spectacle'` à `VARCHAR(60)`.
   (c) Supprimer l'attribut `'langue'` de la table `'Spectacle'`.
   (d) Ajouter une clé étrangère sur l'attribut `'num_spectacle'` de la table `'Comedien'`.

Exercice 24

Nous utiliserons pour cet exercice les requêtes de création des tables d'une base de données pour la gestion des employés d'une société données à la section 3.9.2.

1. Consulter les requêtes données à la section 3.9.2.
2. Créer la base de données `'societe'` puis y accéder.
3. Dans l'onglet SQL, faire un copier/coller ou bien saisir les requêtes données à la section 3.9.2 afin de créer les tables de la base de données.
4. Insérer le tuple suivant dans la table `'Employe'`
   — `(01,'Ali', 'Commercial', '2001-05-05',01)`

Que se passe-t-il ? Pourquoi ?

5. Insérer les tuples suivants dans la table `'Departement'` :
   - `(01, 'Achat')`
   - `(02, 'Administration, Comptabilite et Finance')`
   - `(03,'Production')`
   - `(04, 'Marketing')`
   - `(05, 'Vente')`
6. Insérer les tuples suivants dans la table `'Employe'` :
   - `(01,'Ali', 'acheteur', '2001-05-05',01)`
   - `(02,'Malika', 'chef de departement', '2000-01-03',03)`
   - `(03,'Fouad', 'assistant', '2014-03-30',02)`
   - `(04,'Alia', 'comptable', '1990-05-10',02)`
   - `(05,'Manel', 'commercial', '2015-02-06',05)`
   - `(06,'Raouf', 'chef de produit', '2011-03-20',04)`
7. Modifier le poste de l'employé n ° 1 en "chef de département".
8. Supprimer les employés qui ont été recrutés avant 1995.
9. Ajouter un attribut `'ville'` à la table `'Departement'` puis y insérer la valeur `'Oran'` pour tous les tuples existants.
10. Fusionner les départements `'Marketing'` et `'Vente'` en un seul département ayant le numéro 04.

Exercice 25

Nous utiliserons pour cet exercice les requêtes de création des tables d'une base de données pour la gestion des étudiants données à la section 3.9.3.

1. Consulter les requêtes données à la section 3.9.3.
   Remarque : On considère que pour un étudiant qui n'a pas de note pour un module, il y aura quand même un tuple qui les lie dans la table `'Evaluer'`, avec **`NULL`** qui sera précisé pour l'attribut `'note'`.
2. Créer la base de données `'notes'` puis y accéder.
3. Dans l'onglet SQL, faire un copier/coller ou bien saisir les requêtes données à la section 3.9.3 afin de créer les tables de la base de données et d'y insérer des données.
4. Exécuter les requêtes qui permettent d'afficher les résultats des recherches ci-dessous. Pour chaque requête, le nombre de tuples résultants est indiqué entre parenthèses :
   (a) La liste des prénoms distincts des étudiants. (8 tuples)
   (b) L'intitulé du module dont le code est `'STAT'`. (1 tuple)
   (c) Les étudiants dont le numéro est compris entre 3 et 7. (5 tuples)
   (d) Le nom et le prénom des étudiants dont le nom ou le prénom contient la lettre `'M'`. (4 tuples)
   (e) Les étudiants dont la note du module `'ANA1'` est supérieure à 10. (4 tuples)

(f) Les étudiants dont la note dans un module à coefficient 3 est égale à 12. (1 tuple)

(g) La liste des étudiants ordonnés de manière croissante selon leur note en `'Analyse 1'`. (10 tuples)

(h) Pour chaque étudiant son nom, prénom, et ses notes pour les modules `'ANA1'` et `'ALG1'` (sur la même ligne). (10 tuples)

(i) La liste des étudiants n'ayant pas de notes dans au moins un des modules. (2 tuples)

(j) La liste des étudiants n'ayant aucune note. (1 tuple)

(k) Le nombre d'étudiants n'ayant aucune note. (1 tuple)

(l) La moyenne de notes pour le module `'INF4'`. (1 tuple)

(m) La meilleure note pour chaque module. (3 tuples)

(n) Pour chaque module, son intitulé, sa meilleur note, ainsi que le nom et le prénom des étudiants ayant eu cette meilleure note. (3 tuples)

3.7 Autres exercices

Exercice 26

1. Répondre par vrai ou faux :

   a) La requête CREATE permet de créer une base de données.

   b) La requête DELETE permet de supprimer une base de données.

2. Parmi les requêtes ci-dessous, lesquelles font partie du Langage de Définition de Données (LDD) et lesquelles font partie du Langage de Manipulation de Données (LMD) de SQL :

   a) **`DELETE`**

   b) **`DROP`**

   c) **`CREATE`**

   d) **`UPDATE`**

Exercice 27

Soit la table T suivante :

A	B	C
a1	b1	NULL
a1	b2	c1
a2	b4	c2

1. Quel est l'attribut qui peut jouer le rôle de clé primaire dans la table T ?

2. Donner une requête qui permet de supprimer uniquement le tuple ("a1", "b1", NULL).

3. Donner le résultat de la requête :

```
SELECT T1.A
FROM T AS T1 JOIN T AS T2 ON T1.A=T2.A
```

Exercice 28

Exprimer les mêmes requêtes que celles de l'exercice 20 en SQL.

Exercice 29

Soit le MLD suivant :

— Aéroport(code_aéro, nom, ville)
— Vol(num_vol, aéro_départ*, aéro_destination*, date_départ, heure_départ, date_arrivée, heure_arrivée)

Donner les requêtes SQL qui permettent de :

1. Créer les deux tables "Aéroport" et "Vol".

2. Insérer l'aéroport Atatürk d'Istanbul (code : ISL) dans la base de données.

3. Afficher la liste des villes que l'on peut atteindre à partir la ville d'Oran à travers un vol direct.

Exercice 30

Soit la table $R(debut, fin)$ dans une base de données relationnelle qui représentent les arcs dans un graphe orienté. Chaque tuple (x, y) de R représente un arc du sommet x vers le sommet y.

Donner une requête SQL qui permet de retourner l'extrémité initiale et l'extrémité terminale de tous les chemins de longueur 3 dans le graphe représenté par R. La requête devra retourner une nouvelle table avec les attributs $debut$, fin. Chaque tuple (x, y) obtenu par cette requête devra indiquer qu'il existe dans le graphe un chemin :

$$x \to z \to w \to y$$

pour un certain z et un certain w.

3.8 Solutions des exercices

Solution de l'exercice 22

1.
```
CREATE DATABASE banque;
```

2.
```
CREATE TABLE Client(
  num_client INT,
  nom VARCHAR(20),
  prenom VARCHAR(20),
  ville VARCHAR(20),
  PRIMARY KEY(num_client));
CREATE TABLE Compte(
  num_compte INT,
  num_client INT,
  type VARCHAR(20),
  PRIMARY KEY(num_compte),
  FOREIGN KEY(num_client) REFERENCES Client(num_client));
CREATE TABLE Operation(
  num_op INT,
  num_compte INT,
  montant FLOAT,
  informations VARCHAR(100),
  PRIMARY KEY(num_op),
  FOREIGN KEY(num_compte) REFERENCES Compte(num_compte));
```

3. Il n'est pas possible de supprimer la table `Client` car c'est une table père par rapport à la table `Compte` à travers la clé étrangère `num_client`.

4.
```
DROP TABLE Operation;
```

Solution de l'exercice 23

4. (a)
```
ALTER TABLE Programmer
ADD heure_debut TIME;
```

   (b)
```
ALTER TABLE Spectacle
MODIFY titre VARCHAR(60);
```

   (c)
```
ALTER TABLE Spectacle
DROP langue;
```

   (d)
```
ALTER TABLE Comedien
ADD FOREIGN KEY(num_spectacle)
  REFERENCES Spectacle(num_spectacle);
```

Solution de l'exercice 24

4. Le tuple ne peut pas être inséré car le département n ° 1 n'a pas encore été créé dans la table `Departement`.

5.
```
INSERT INTO Departement VALUES
  (01, 'Achat'),
  (02, 'Administration, Comptabilite et Finance'),
  (03,'Production'),
  (04, 'Marketing'),
  (05, 'Vente');
```

6.
```
INSERT INTO Employe VALUES
  (01,'Ali', 'acheteur', '2001-05-05',01),
  (02,'Malika', 'chef de departement', '2000-01-03',03),
  (03,'Fouad', 'assistant', '2014-03-30',02),
  (04,'Alia', 'comptable', '1990-05-10',02),
  (05,'Manel', 'commercial', '2015-02-06',05),
  (06,'Raouf', 'chef de produit', '2011-03-20',04);
```

7.
```
UPDATE Employe
SET poste='chef de departement'
WHERE num_emp=1;
```

8.
```
DELETE FROM Employe
WHERE date_recrutement < '1995-01-01';
```

9.
```
ALTER TABLE Departement
ADD ville VARCHAR(20);
UPDATE Departement
SET ville='Oran';
```

10.
```
UPDATE Employe
SET num_dept=04
WHERE num_dept=05;
DELETE FROM Departement
WHERE num_dept=05;
UPDATE Departement
SET nom_dept="Marketing et vente"
WHERE num_dept=4;
```

Solution de l'exercice 25

4. (a)
```
SELECT DISTINCT prenom
FROM Etudiant;
```

(b)
```
SELECT intitule
FROM Module
WHERE code_module='STAT';
```

(c)
```
SELECT *
FROM Etudiant
WHERE num_etu>=3 AND num_etu<=7;
```

(d)
```
SELECT nom, prenom
FROM Etudiant
WHERE nom LIKE '%M%' OR prenom LIKE '%M%';
```

(e)
```
SELECT Etudiant.*
FROM Etudiant NATURAL JOIN Evaluer
WHERE code_module='ANA1' AND note>10;
```

(f)
```
SELECT Etudiant.*
FROM Etudiant NATURAL JOIN Evaluer NATURAL JOIN Module
WHERE coefficient=3 AND note=12;
```

(g)
```
SELECT Etudiant.*
FROM Etudiant NATURAL JOIN Evaluer NATURAL JOIN Module
WHERE intitule='Analyse 1'
ORDER BY note ASC;
```

(h)
```
SELECT nom, prenom, eva1.note AS note_ANA1,
  eva2.note AS note_ALG1
FROM (Etudiant
  JOIN Evaluer AS eva1
    ON (Etudiant.num_etu=eva1.num_etu
      AND eva1.code_module='ANA1'))
  JOIN Evaluer AS eva2
    ON (Etudiant.num_etu=eva2.num_etu
      AND eva2.code_module='ALG1')
```

(i)
```
SELECT DISTINCT Etudiant.*
FROM Etudiant NATURAL JOIN Evaluer
WHERE note IS NULL;
```

(j)
```
SELECT *
FROM Etudiant
WHERE num_etu NOT IN (
  SELECT Etudiant.num_etu
  FROM Etudiant NATURAL JOIN Evaluer
  WHERE note IS NOT NULL);
```

(k)
```
SELECT COUNT(*)
FROM Etudiant
WHERE num_etu NOT IN (
    SELECT num_etu
    FROM Evaluer
    WHERE note IS NOT NULL);
```

(l)
```
SELECT AVG(note)
FROM Evaluer
WHERE code_module='INF4;
```

(m)
```
SELECT intitule, MAX(note)
FROM Evaluer NATURAL JOIN Module
GROUP by code_module;
```

(n)
```
SELECT intitule, nom, prenom, note
FROM Etudiant NATURAL JOIN Evaluer NATURAL JOIN Module
WHERE (code_module, note) IN
    (SELECT code_module, MAX(note)
    FROM Evaluer
    GROUP BY code_module);
```

Solution de l'exercice 26

1. a) Vrai.

   b) Faux.

2. Requête qui fait parti du LDD : **CREATE**,

   Requêtes qui font parti du LMD : **DELETE**, **DROP**, **UPDATE**.

Solution de l'exercice 27

1. L'attribut qui peut jouer le rôle de clé primaire dans la table T est l'attribut B car chaque tuple de la table possède une valeur pour cet attribut et il n'y a pas deux tuples avec la même valeur pour cet attribut.

2. Exemple de requêtes possibles :
   - `DELETE FROM R WHERE B="b1";`
   - `DELETE FROM R WHERE C IS NULL;`

3. Résultat de la requête :

R1.A
a1
a1
a1
a1
a2

Solution de l'exercice 28

1.
```
SELECT *
FROM Site
WHERE ville_site='Tiaret';
```

2.
```
SELECT telephone_guide
FROM Site NATURAL JOIN Visite NATURAL JOIN Guide
WHERE ville_site='Ghardaia';
```

3.
```
SELECT nom_guide, prenom_guide
FROM Site NATURAL JOIN Visite NATURAL JOIN Guide
WHERE ville_site != ville_guide;
```

4.
```
SELECT nom_guide, prenom_guide
FROM Guide
WHERE id_guide NOT IN
  SELECT Guide.id_guide
  FROM Site NATURAL JOIN Visite NATURAL JOIN Guide
  WHERE ville_site = ville_guide;
```

Solution de l'exercice 29

1.
```
CREATE TABLE Aeroport(
  code_aero CHAR(3),
  nom VARCHAR(30),
  ville VARCHAR(30),
  PRIMARY KEY(code_aero));

CREATE TABLE Vol(
  num_vol INT,
  aero_depart CHAR(3),
  aero_destination CHAR(3),
  date_depart DATE,
  heure_depart TIME,
  date_arrivee DATE,
  heure_arrivee TIME,
  PRIMARY KEY(num_vol),
  FOREIGN KEY(aero_depart)
    REFERENCES Aeroport(code_aero),
  FOREIGN KEY(aero_destination)
    REFERENCES Aeroport(code_aero));
```

2.
```
INSERT INTO Aeroport
VALUES("IST","Aeroport d'Istanbul","Istanbul");
```

3.
```
SELECT A2.ville
FROM Aeroport AS A1
```

```
  JOIN Vol ON A1.code_aero=aero_depart
  JOIN Aeroport AS A2 ON A2.code_aero=aero_destination
WHERE A1.ville="Oran"
```

Solution de l'exercice 30

```
SELECT R1.debut AS debut, R3.fin AS fin
FROM R AS R1
  JOIN R AS R2 ON R1.fin=R2.debut
  JOIN R AS R3 ON R2.fin=R3.debut;
```

3.9 Annexe

Ici sont données les requêtes données à exécuter pour chaque exercice précisé.

3.9.1 Requêtes pour l'exercice 23

```
CREATE TABLE Spectacle(
  num_spectacle INT,
  titre VARCHAR(40),
  lieu VARCHAR(20),
  langue VARCHAR(20),
  PRIMARY KEY(num_spectacle)
);
CREATE TABLE Comedien(
  num_comedien INT,
  nom VARCHAR(20),
  prenom VARCHAR(20),
  num_spectacle INT,
  PRIMARY KEY(num_comedien)
);
CREATE TABLE Programmer(
  date DATE,
  num_spectacle INT,
  tarif FLOAT,
  PRIMARY KEY(date,num_spectacle),
  FOREIGN KEY(num_spectacle) REFERENCES Spectacle(num_spectacle)
);
```

3.9.2 Requêtes pour l'exercice 24

```
CREATE TABLE Departement(
  num_dept INT,
  nom_dept VARCHAR(40),
```

```
  PRIMARY KEY(num_dept));
CREATE TABLE Employe(
  num_emp INT,
  nom_emp VARCHAR(20),
  poste VARCHAR(20),
  date_recrutement DATE,
  num_dept INT,
  PRIMARY KEY(num_emp),
  FOREIGN KEY(num_dept) REFERENCES Departement(num_dept));
```

3.9.3 Requêtes pour l'exercice 25

```
CREATE TABLE Etudiant(
  num_etu INT,
  nom VARCHAR(20),
  prenom VARCHAR(20),
  PRIMARY KEY(num_etu));
CREATE TABLE Module(
  code_module CHAR(4),
  intitule VARCHAR(25),
  coefficient INT,
  PRIMARY KEY(code_module));
CREATE TABLE Evaluer(
  num_etu INT,
  code_module CHAR(4),
  note FLOAT,
  PRIMARY KEY(num_etu,code_module),
  FOREIGN KEY(num_etu) REFERENCES Etudiant(num_etu),
  FOREIGN KEY(code_module) REFERENCES Module(code_module));
INSERT INTO Etudiant VALUES
  (01, 'Bouanani', 'Malika'),
  (02, 'Berrahou', 'Ilies'),
  (03, 'Bendouma', 'Racim'),
  (04, 'Kharroubi', 'Yacine'),
  (05, 'Kazi Tani', 'Leyth'),
  (06, 'Bendoukha', 'Manel'),
  (07, 'Tlitli', 'Younes'),
  (08, 'Zerhouni', 'Yacine'),
  (09, 'Bahri', 'Katia'),
  (10, 'Badis', 'Malika');
INSERT INTO Module VALUES
  ('ANA1','Analyse 1',6),
  ('ALG1','Algebre 1', 3),
  ('STAT', 'Probablites-satistques',3);
INSERT INTO Evaluer VALUES
  (01, 'ANA1', 02.5),
  (01, 'ALG1', 12),
  (01, 'STAT', 9),
```

```
(02, 'ANA1', 19),
(02, 'ALG1', 15),
(02, 'STAT', 8),
(03, 'ANA1', 9),
(03, 'ALG1', 9),
(03, 'STAT', 9),
(04, 'ANA1', NULL),
(04, 'ALG1', 8),
(04, 'STAT', 11),
(05, 'ANA1', 2),
(05, 'ALG1', 0),
(05, 'STAT', 5),
(06, 'ANA1', 16),
(06, 'ALG1', 14),
(06, 'STAT', 10),
(07, 'ANA1', 16),
(07, 'ALG1', 16),
(07, 'STAT', 16),
(08, 'ANA1', 12.5),
(08, 'ALG1', 14),
(08, 'STAT', 17),
(09, 'ANA1', 03),
(09, 'ALG1', 9),
(09, 'STAT', 10),
(10, 'ANA1', NULL),
(10, 'ALG1', NULL),
(10, 'STAT', NULL);
```

BIBLIOGRAPHIE

[1] Dominique Nanci, Bernard Espinasse, Bernard Cohen, and Bernard Cohen. Ingénierie des systèmes d'information : Merise : deuxième génération. Vuibert Paris, 2001.

[2] C. J. Date. An Introduction to Database Systems (4th edition). Addison-Wesley Publishing Co., 1987.

[3] J. R. Rumble and F. J. Smith. Database systems in science and engineering. CRC Press, 1990.

[4] A. Gamache. Architectures, modèles et langages de données, 2005.

[5] M. Clouse. Algèbre relationnelle : guide pratique de conception d'une base de données relationnelle normalisée. Editions ENI, 2008.

[6] J. R. Groff and P. N. Weinberg. The complete reference SQL, third edition. McGraw-Hill, 2009.