جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم والتقنيات
FACULTÉ DES SCIENCES ET TECHNIQUES

Bases de Données

Par :
Pr. Imad Zeroual

Faculté des Sciences et Techniques – Errachidia / 2019-2020

Plan

Introduction

Définitions, Modèles du SGBD.

Conception d’une base de données

Entité-Associations, Normalisation.

Modèle logique de données (Relationnel)

Du E/A au Modèle relationnel, L’algèbre relationnelle.

Langage de manipulation des données SQL

Commandes, Fonctions.

Conclusion

Résumé général, Références.

Introduction

Définitions, Modèles du SGBD.

01

‘Base’ Dictionnaire Français

Partie inférieure d'un objet ou sous l'objet, sur laquelle il repose.
Synonymes : pied, point…

Donnée fondamentale ou principe sur lequel repose une pensée, un système.
Synonymes : source, origine, racine, fondement, principe…

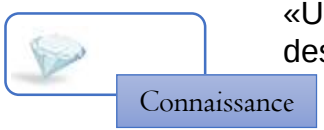
Substance qui réagit avec un acide et le neutralise pour former un sel.
Domaine : Chimie.

Chacun des côtés parallèles d'un polygone, coté d'un triangle ou face d'un cône opposé au sommet.
Domaine : Mathématique.

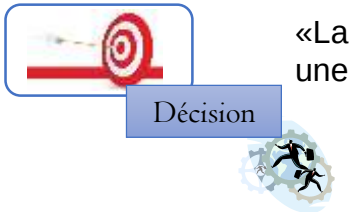
‘ Donnée ’

«Une donnée est un élément brut, qui n’a pas encore été interprété, mis en contexte.»

«Une information est par définition une donnée interprétée.»



«Une action réalisée en fonction des connaissances disponibles.»



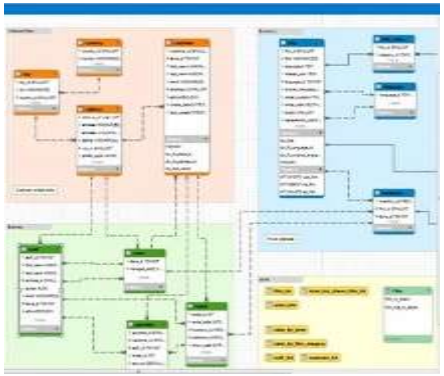
«La connaissance comme une information comprise.»

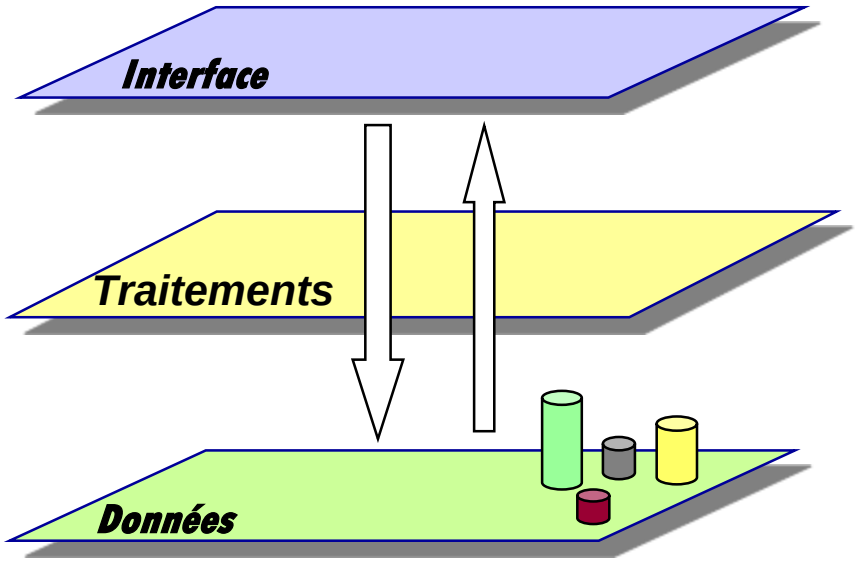
‘Système d’information ’

Système constitué des ressources humaines, des ressources matérielles et des procédures permettant *d'acquérir*, de *stocker*, de *traiter* et de *diffuser* les éléments d'information pertinents au fonctionnement d'une organisation »

Base de Donnée

Les dimensions d'une application ?





Stations de travail

Langages de programmation
Manipulation de données

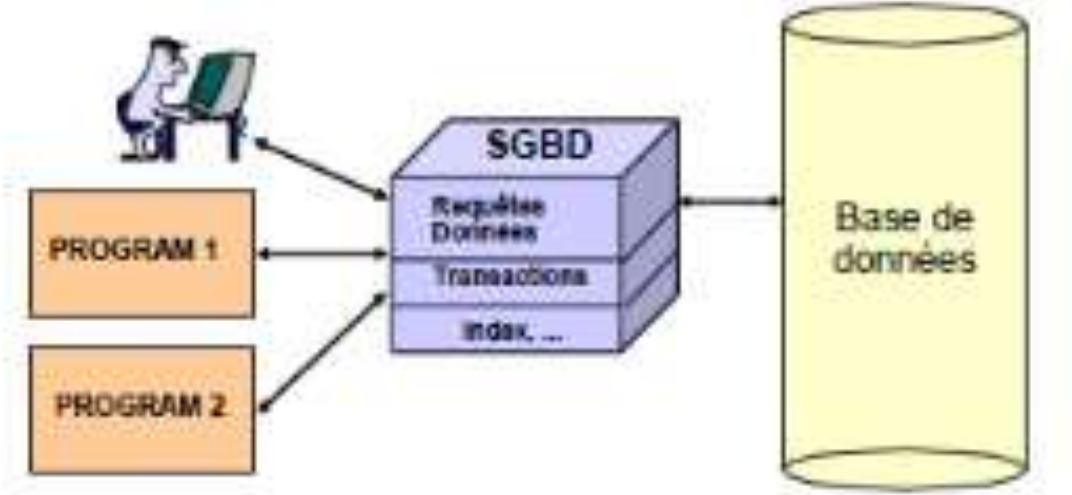
SGBD
Bases de Données

‘ Base de Donnée ’

- ✓ BD est un ensemble **structuré** de données, **enregistrées** sur des supports accessibles par **l’ordinateur** pour satisfaire simultanément plusieurs **utilisateurs** de manière **sélective** en un **temps opportun**.
- ✓ BD est un ensemble de données **reliées** entre-elles de manière **logique**.
- ✓ Tout système d’information est **construit autour** de bases de données !!

‘SGBD ,

Un **S**ystème de **G**estion de **B**ases de **D**onnées **SGBD** est un ensemble de logiciels (Programmes) permettant aux utilisateurs de définir, créer, maintenir, contrôler et accéder à la **Base de Données**.

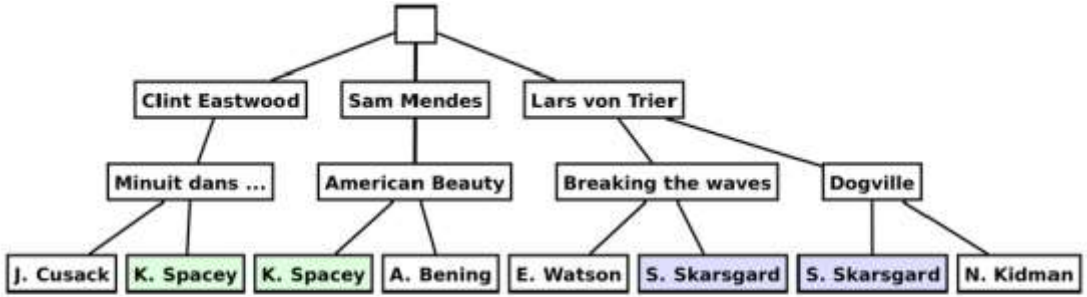


‘SGBD’

Exemples :

‘ Modèles du SGBD ’

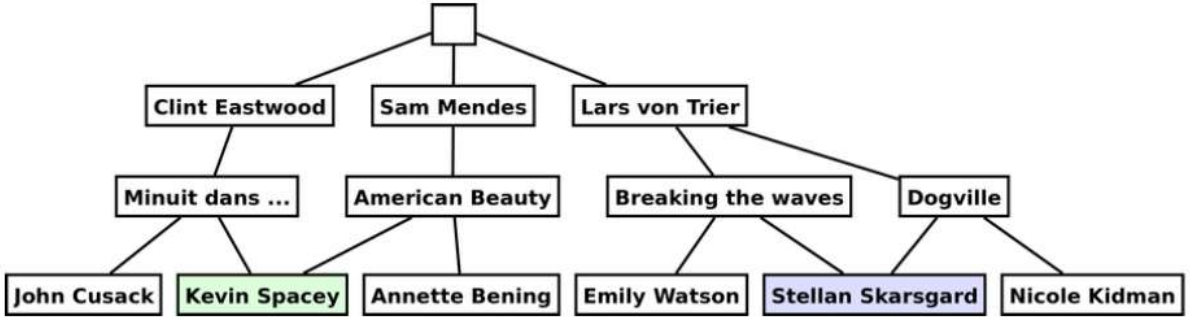
Hiérarchique :



- Développé par IBM pour le programme spatial Apollo de la NASA (années 1960)
- Lie des enregistrements dans une structure arborescente de façon à ce que chaque enregistrement n'ait qu'un seul possesseur
- Liaisons de type *1 vers N*

‘ Modèles du SGBD ’

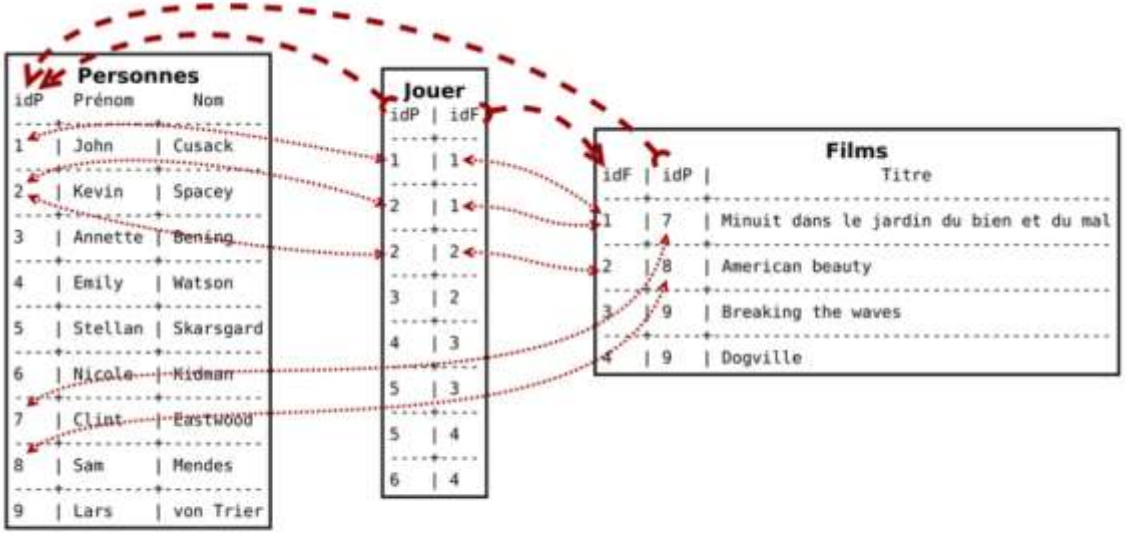
Réseaux :



- Permet de lever de nombreuses difficultés du modèle hiérarchique
- Possibilité d'établir des liaisons de type *N vers N*
- Programme dépendant de la structure de données (impose de connaître les chemins des liens)

‘Modèles du SGBD’

Relationnel :

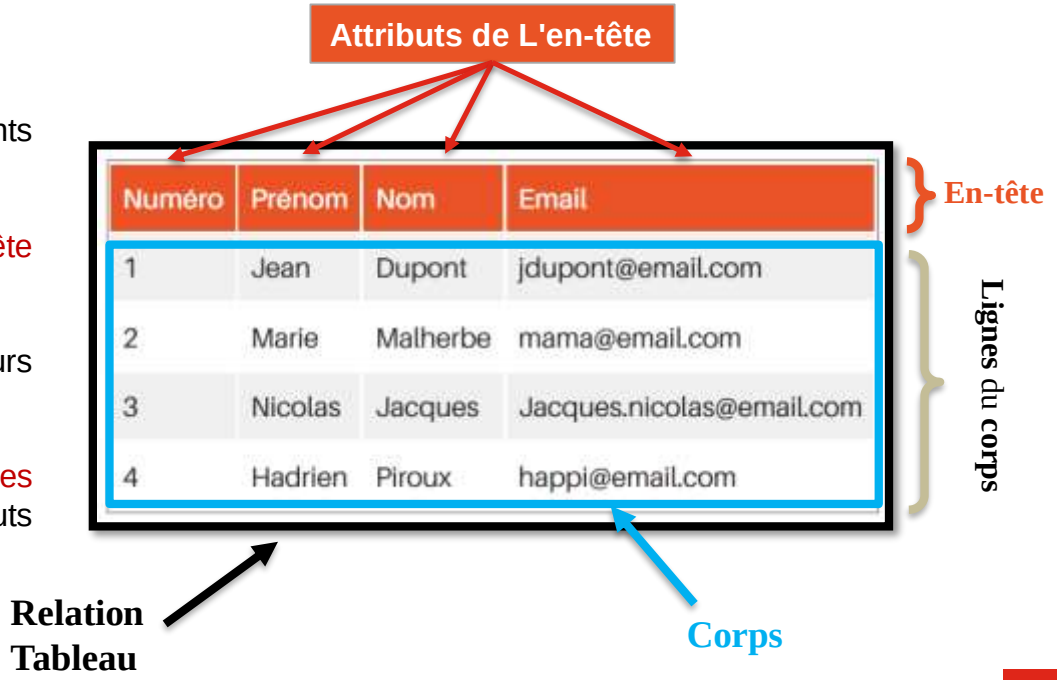


- 1970 : Codd publie un article où il propose de stocker des données hétérogènes dans des tables

Modèles du SGBD

Relationnel :

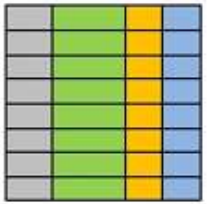
- ✓ Les données sont représentées dans différents relations/tableaux pouvant être liés entre eux.
- ✓ Une relation est composée de deux parties, l'en-tête et le corps.
- ✓ L'en-tête est lui-même composé de plusieurs attributs.
- ✓ Quant au corps, il s'agit d'un ensemble de lignes composées d'autant d'éléments qu'il y a d'attributs dans le corps.



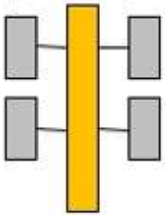
‘ Modèles du SGBD ’

Autres types :

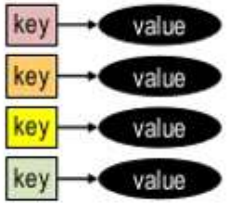
Relational



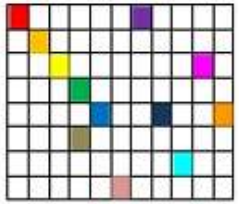
Analytical (OLAP)



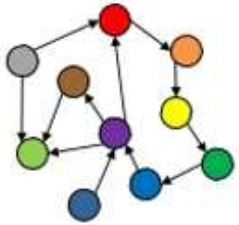
Key-Value



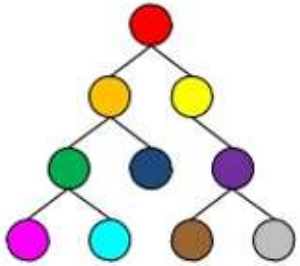
Column-Family



Graph



Document



‘ Pratique ’

Ex-1 :

1. Les besoins que doit satisfaire une base de données sont :

- Description
- ✓ Manipulation
- Ergonomie
- Respect des auteurs
- ✓ Efficacité
- Exactitude
- ✓ Garantie contre la perte
- ✓ Confidentialité

‘ Pratique ,

Ex-1 :

2. Un système d'information d'une entreprise est composé des :

- Ordinateurs
- Opérateurs de saisie
- ✓ Ressources humaines, des ressources matérielles et des procédures

‘ Pratique ,

Ex-1 :

3. Sont des types de SGBD :

- ✓ - Réseau
- Anneau
- Objet
- ✓ - Relationnel
- ✓ - Hiérarchique

‘ Pratique ,

Ex-1 :

4. Dans le modèle SGBD relationnel, une relation est composée de trois parties, l'en-tête, le corps et le pied :

- Vrai

✓ - Faux

Numéro	Prénom	Nom	Email
1	Jean	Dupont	jdupont@email.com
2	Marie	Malherbe	mama@email.com
3	Nicolas	Jacques	Jacques.nicolas@email.com
4	Hadrien	Piroux	happi@email.com

En-tête

Corps

‘ Pratique ’

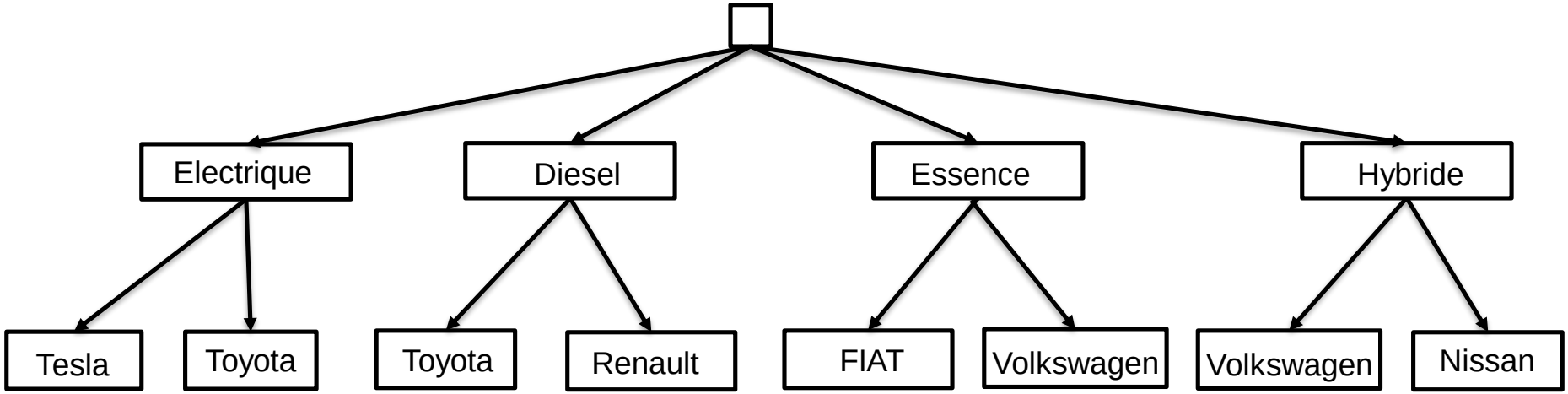
Ex-1 :

5. Sont des logiciels de SGBD :

- ✗ - ~~MiSQL~~ MySQL
- ✗ - ~~Microsoft Office~~ Microsoft Access ou Microsoft SQL Server
- ✓ - Oracle
- ✓ - PostgreSQL
- ✗ - ~~SamDB~~
- ✓ - MariaDB
- ✓ - SQLite
- ✗ - ~~Freefire~~ Firebird

‘ Pratique ,

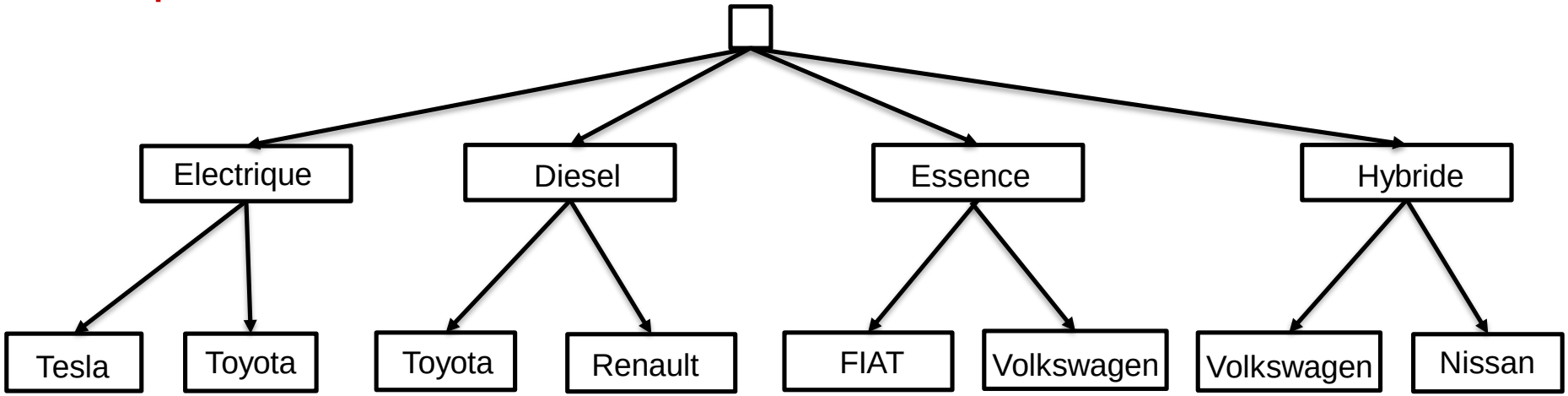
Ex-2 :



1. Dans quel modèle de SGBD les données sont-elles enregistrées de cette façon ? Justifier ?
2. Proposez un autre modèle de SGBD pour stocker ces enregistrements ?

‘ Pratique ’

Rep-2 :



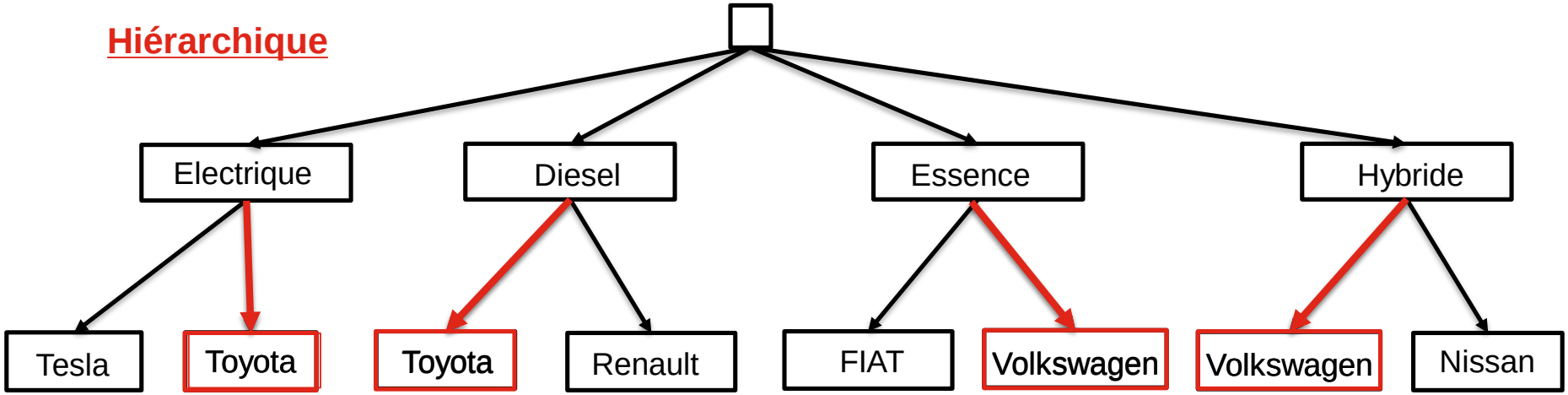
1. C’est le modèle **hiérarchique,** car les enregistrements sont liées dans une **structure arborescente** de façon à ce que chaque enregistrement n’ait qu’un seul processeur (**type 1 vers N**).

‘Pratique’

Rep-2 :

2. Le modèle **Réseaux,** où les enregistrements sont liées dans **structure arborescente** avec une possibilité d’établir des liaisons de type **N vers N.**

Hiérarchique

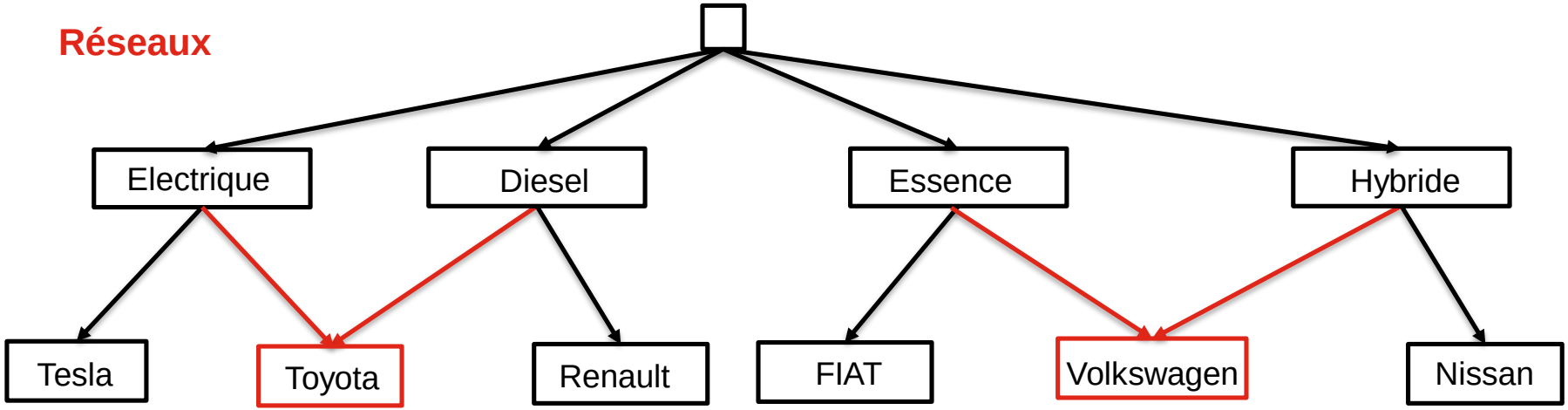


‘ Pratique ,

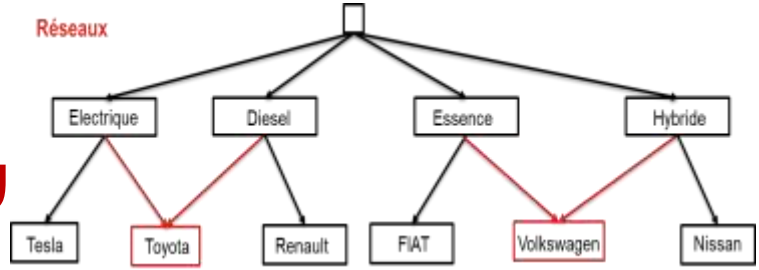
Rep-2 :

2. Le modèle **Réseaux,** où les enregistrements sont liées dans **structure arborescente** avec une possibilité d’établir des liaisons de type **N vers N.**

Réseaux



‘ Pratique ’



Rep-2 :

2. Dans le modèle **Relationnel,** les enregistrements sont représentées sous forme de **Tables.**

Constructeurs automobile		
Id_C	Nom	Pays
C1	Tesla	USA
C2	Toyota	Japon
C3	Volkswagen	Allemagne
C4	FIAT	Italie
C5	Nissan	Japon
C6	Renault	France
C7	Hyundai	Corée du Sud
C8	…	…

Fabriquer	
Id_C	Id_M
C1	M1
C2	M1
C2	M2
C3	M3
C3	M4
C4	M3
C5	M4
C6	M2

Moteurs	
Id_M	Nom
M1	Electrique
M2	Diesel
M3	Essence
M4	Hybride
M5	…

Conception de BD

02

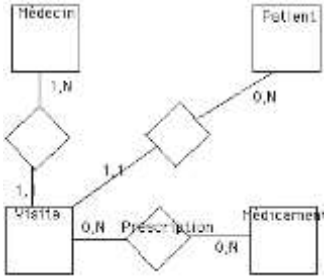
Entité-Associations, Normalisation.

‘Modélisation des données’

La modélisation des données est l'analyse et la conception de l'information contenue dans le système d'information.

Pour construire une base de données, il faut :

1. Construire un schéma conceptuel, modélisé sous forme d’entités et d’associations ;
2. transformer le schéma E/A en schéma relationnel ;
3. Mettre en œuvre via un SGBD.



‘ Entité/ Association ’

- ✓ Le modèle **E/A** est un Formalisme **graphique** pour la modélisation de données.
- ✓ Origine : Travaux de Peter Chen, 1976, USA.
- ✓ Succès dus à :
  - ➢ Langage graphique
  - ➢ Concepts simples :
    - ▪ Choses (objets) → entités
    - ▪ Liens entre les choses (objets) → association
    - ▪ Regroupement des choses de même nature : classes d’entités, classes d’association.

‘ Entité/ Association ’

Entité :

- "une chose" qui existe et qui peut être distinguée de façon unique.
- abstraite ou concrète

Exemple :

Personne

Voiture

Produit

‘ Entité/ Association ’

Attribut :

- propriété d’une entité.
- prend des valeurs simples, par exemple entiers ou chaînes de caractères (**domaine d'attribut**)

Exemple :

Entité

Personne

- ✓ Id_Personne
- ✓ Nom
- ✓ Prénom
- ✓ Age
- ✓ Adresse
- ✓ NumTél

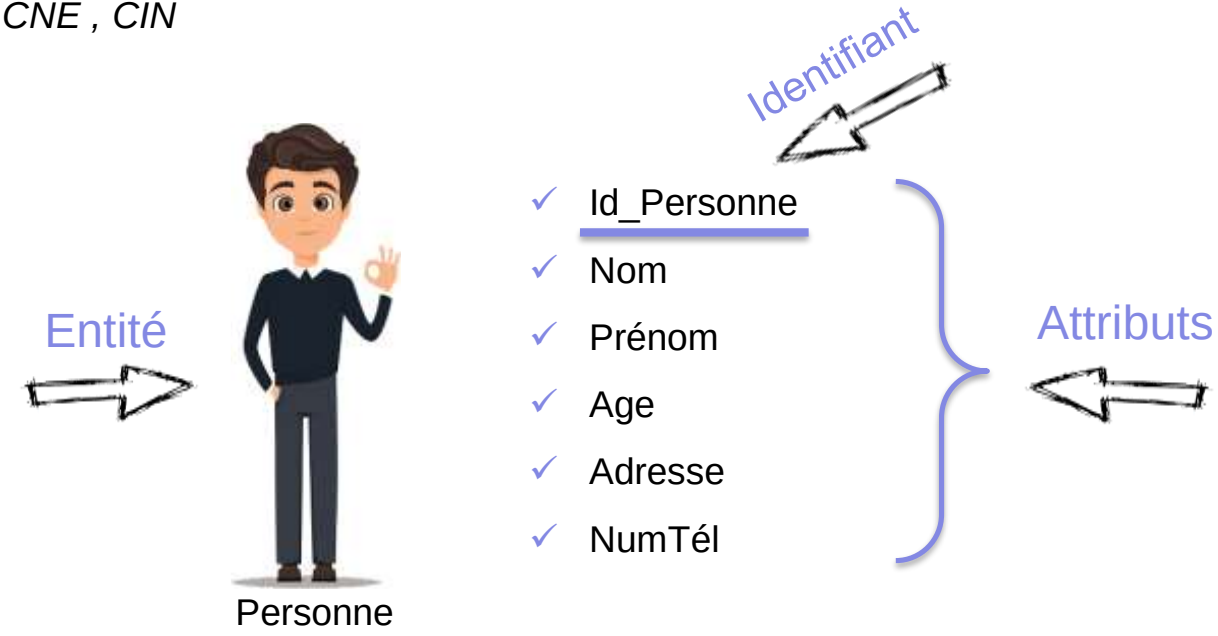
Attributs

‘ Entité/ Association ’

Identifiant **/** Clé primaire :

- C’est la propriété qui identifie de façon unique chaque occurrence d’entité.
  - *Ex.: CNE , CIN*

Exemple 1 :

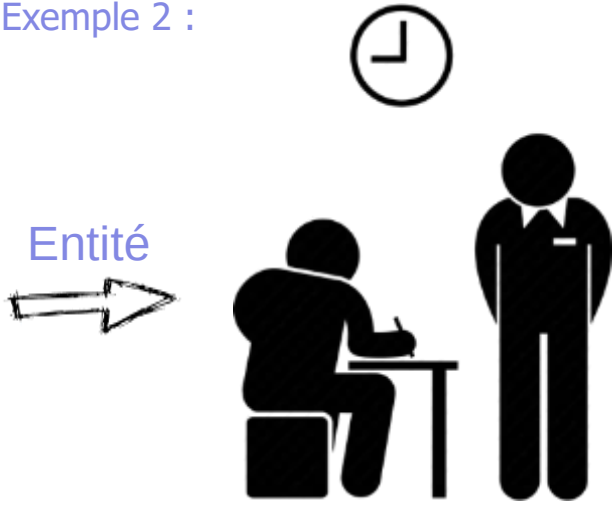


‘ Entité/ Association ’

Identifiant **/** Clé primaire :

- C’est la propriété qui identifie de façon unique chaque occurrence d’entité.
  - *Ex.: CNE , CIN*

Exemple 2 :



- ✓ Code_module
- ✓ Nom_Filière
- ✓ Titre_cours
- ✓ Date
- ✓ Local

Identifiant

Attributs

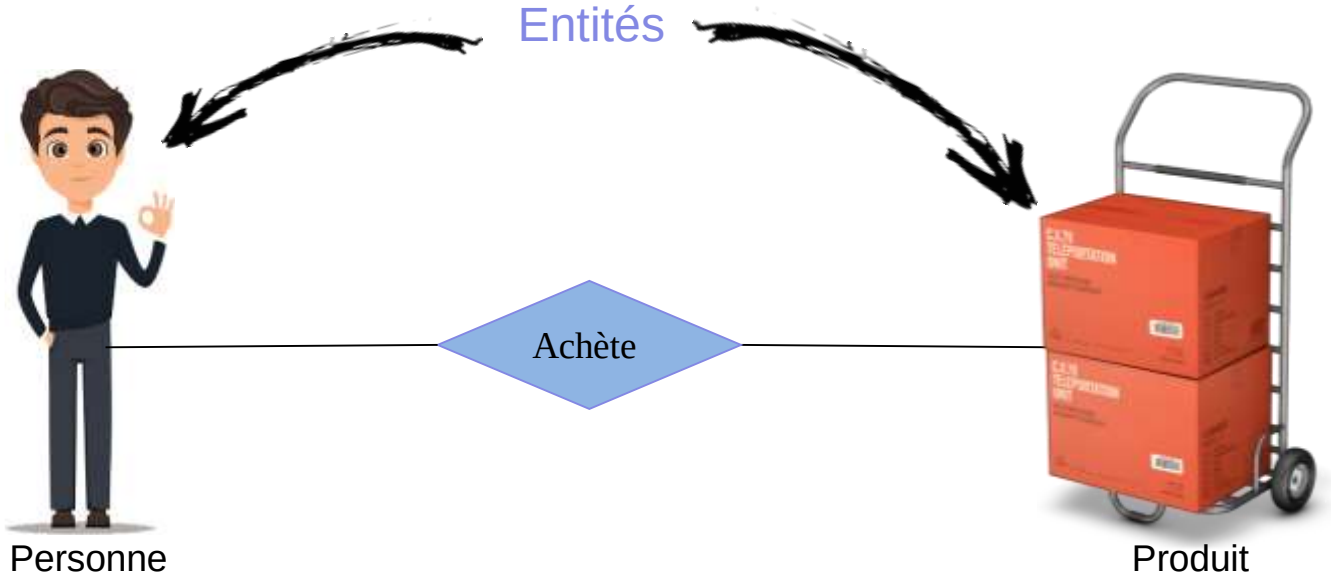
Il y a qu'une **seule clé primaire** par table. Par contre, il est possible de créer une clé primaire à partir de deux champs ou plus. d'une table.

‘ Entité/ Association ’

Association (Relation) :

- C’est un lien entre deux ou plusieurs entités.

Exemple : *une personne achète un produit.*



‘ Entité/ Association ’

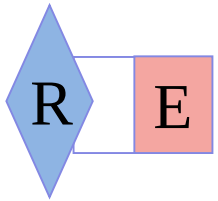
Association (Relation) :

- C’est un lien entre deux ou plusieurs entités.

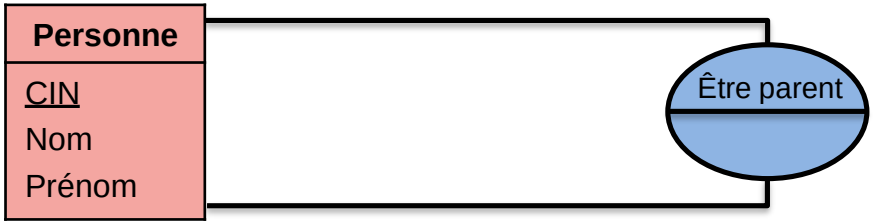
Degré d’une Association

Si **K** est le degré d’une association :

✓ K = 1 : relation unaire (ou récursive)



Exemple :



‘ Entité/ Association ’

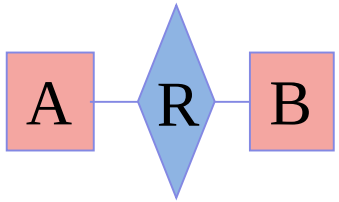
Association (Relation) :

- C’est un lien entre deux ou plusieurs entités.

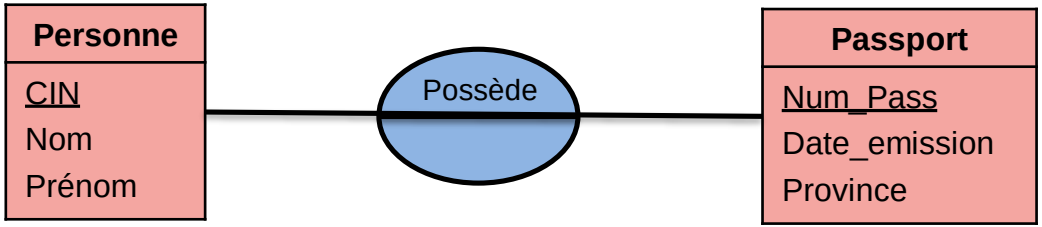
Degré d’une Association

Si **K** est le degré d’une association :

✓ K = 2 : relation binaire



Exemple :



‘ Entité/ Association ’

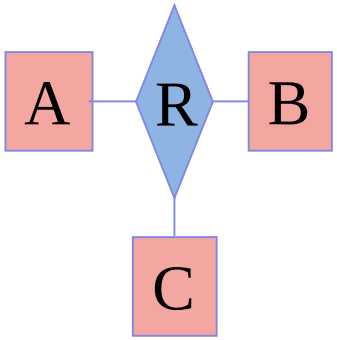
Association (Relation) :

- C’est un lien entre deux ou plusieurs entités.

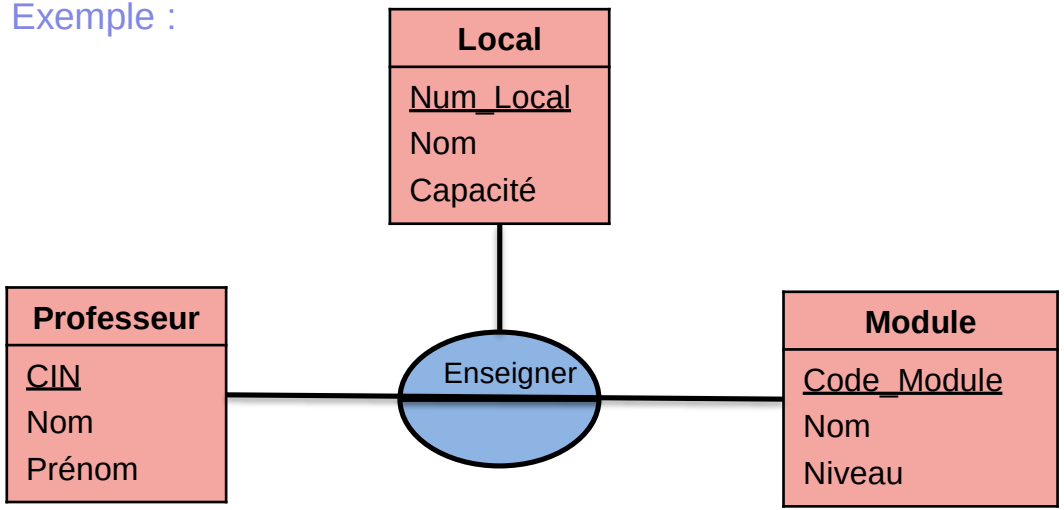
Degré d’une Association

Si **K** est le degré d’une association :

✓ K = 3 : relation ternaire



Exemple :

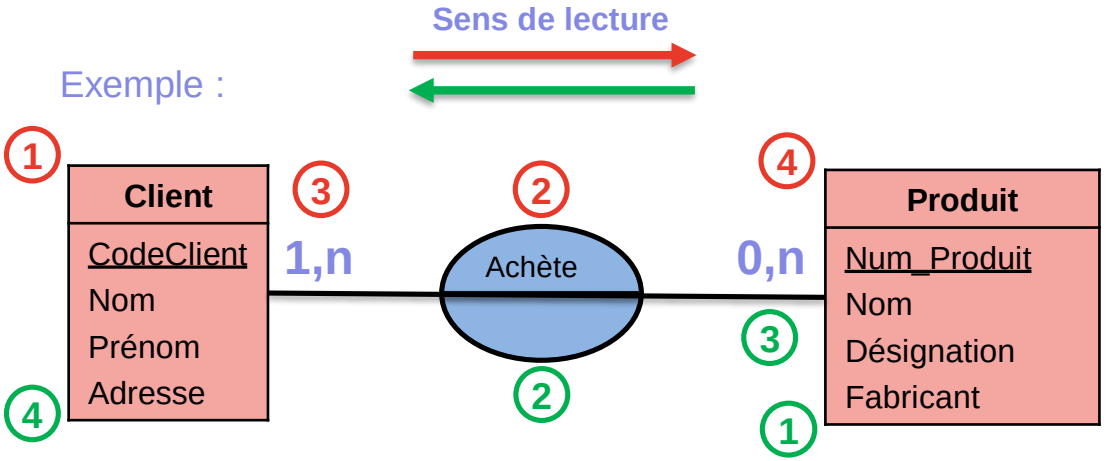


‘ Entité/ Association ’

Cardinalité :

- Mesure le degré de participation de l’entité à l’association.
- Nous avons **4** possibilités :
  - **0,1** : au moins zéro, au plus 1
  - **0,n** : au moins zéro, au plus n
  - **1,1** : au moins 1, au plus 1
  - **1,n** : au moins 1, au plus n
- Cardinalités minimales : 0 et 1
- Cardinalités maximales : 1 et n

Exemple :



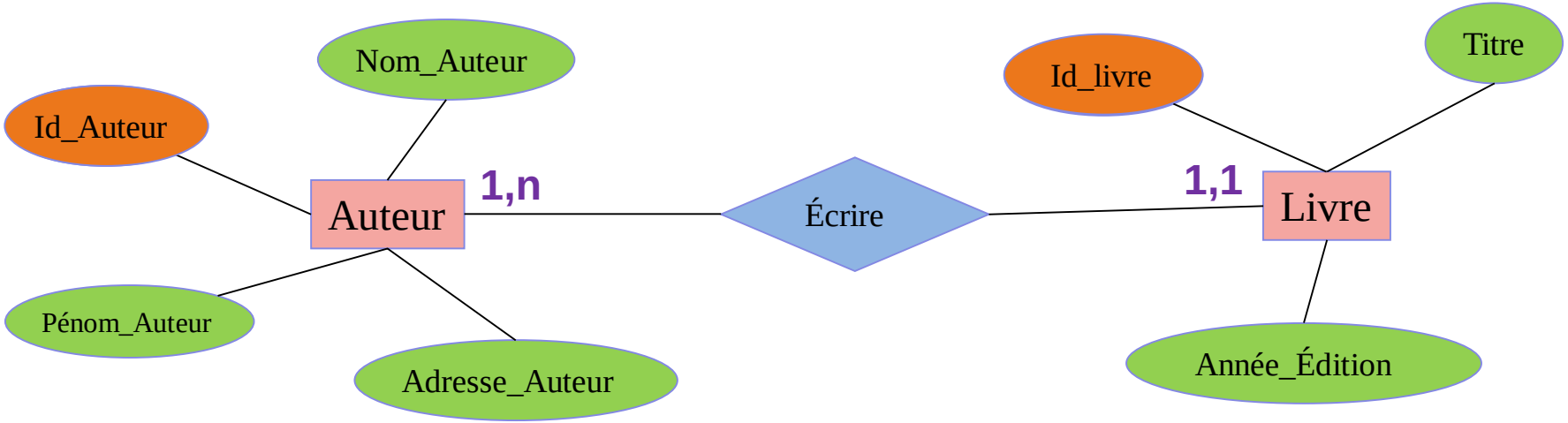
✓ Un client achète **un ou plusieurs** produits.

✓ Un produit est acheté par **aucun ou plusieurs** client.

‘ Entité/ Association ’

Résumé :

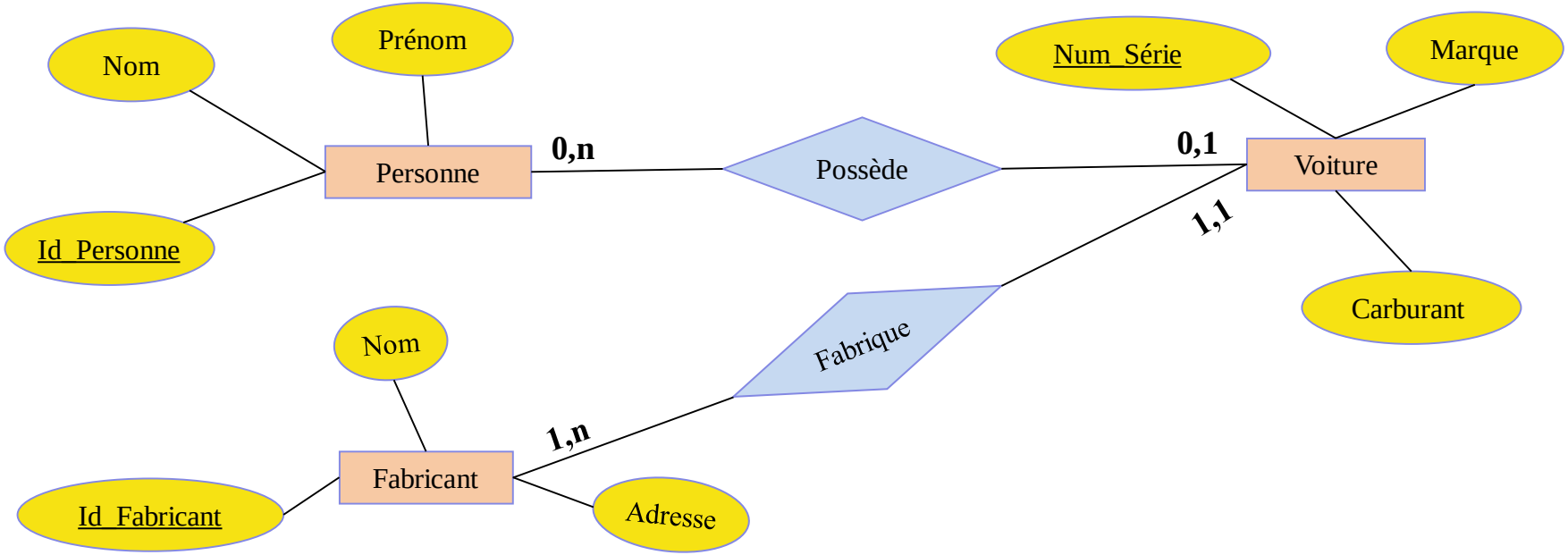
Entités Attributs Identifiants Association Cardinalités



‘ Pratique ’

Ex-1 :

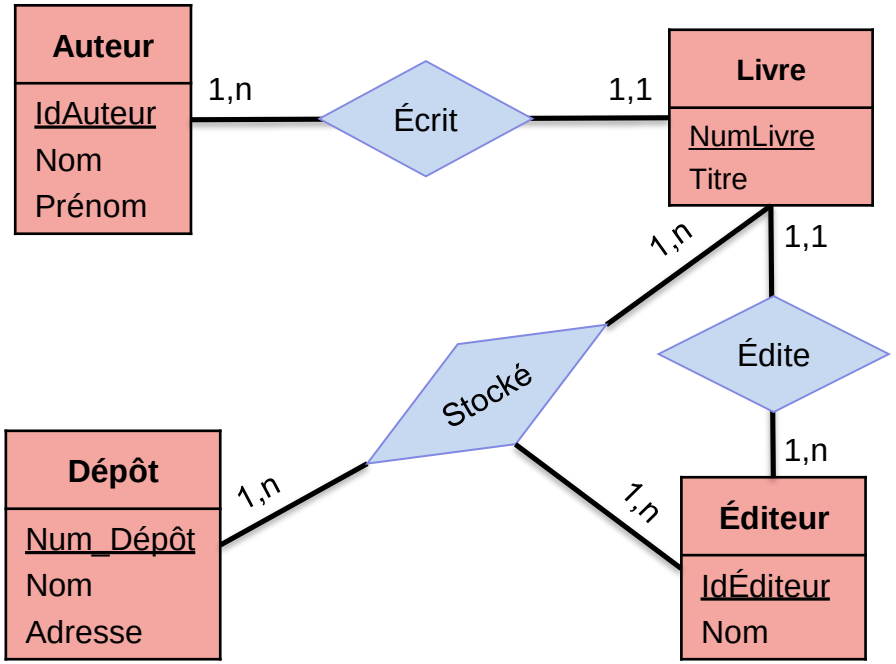
Proposer un diagramme du modèle Entité-Association représentant la relation entre Personne, Voiture, et Fabricant ?



‘ Pratique ’

Ex-2 :

À partir de ce diagramme, répondre aux questions suivantes :



1. Est-il possible d’avoir des auteurs homonymes ?

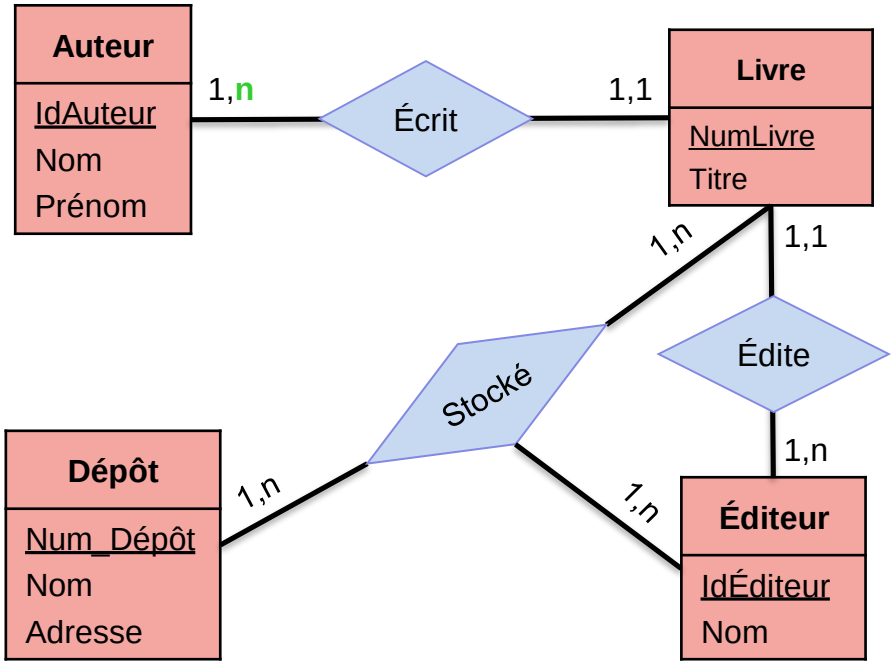
Réponse :

Oui, car le nom n’identifie pas les auteurs. Il peut donc y avoir des homonymes.

‘ Pratique ’

Ex-2 :

À partir de ce diagramme, répondre aux questions suivantes :



2. Un auteur peut-il écrit plusieurs livre ?

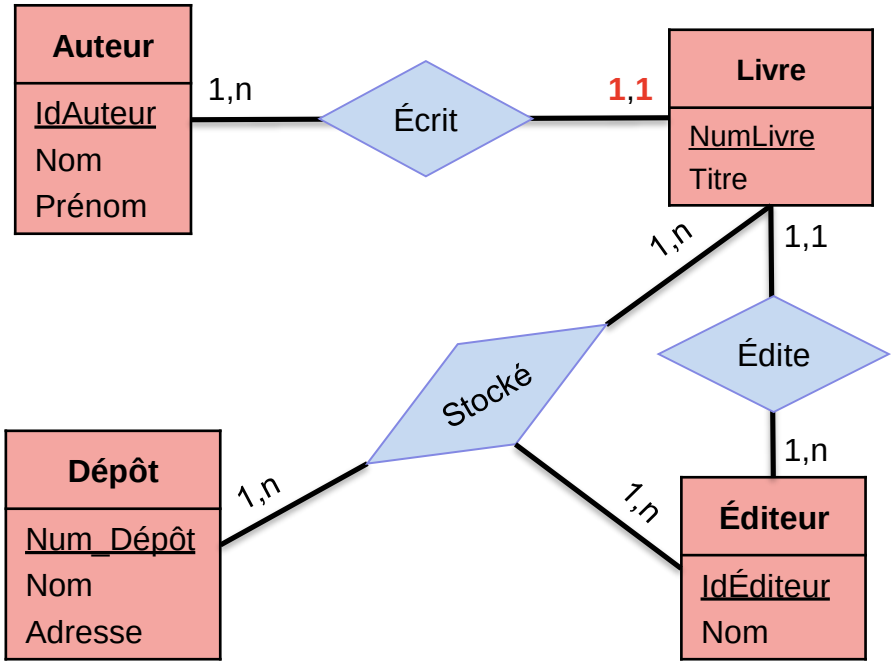
Réponse :

Oui, il peut le faire car sa cardinalité maximale vaut n.

‘ Pratique ’

Ex-2 :

À partir de ce diagramme, répondre aux questions suivantes :



3. Un livre peut-il correspondre à plusieurs auteurs ?

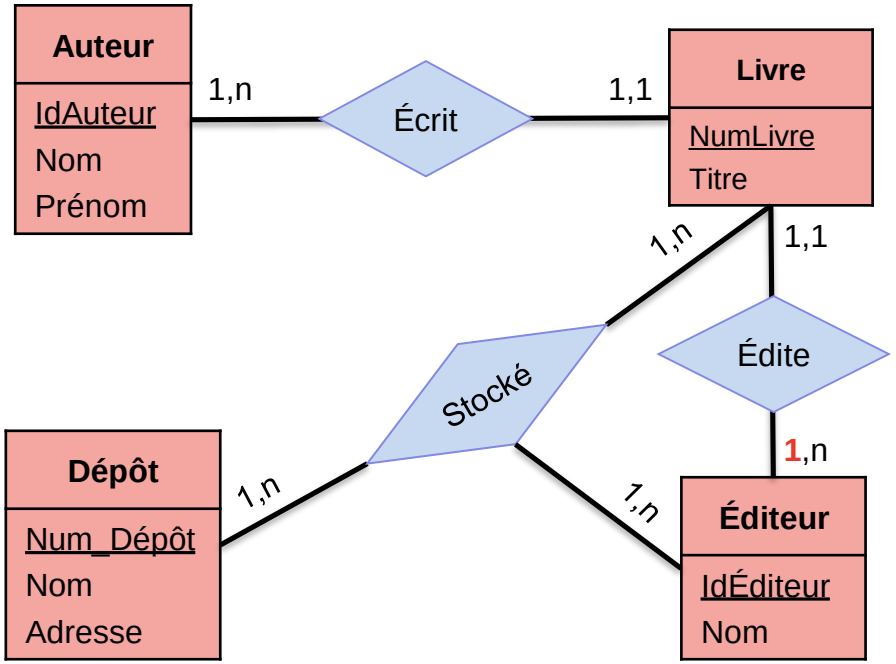
Réponse :

Non, chaque livre correspond à un et un seule auteur, car sa cardinalité minimale et maximale valent 1.

‘ Pratique ’

Ex-2 :

À partir de ce diagramme, répondre aux questions suivantes :



4. Est-il possible qu'un éditeur ne publie aucun livre ?

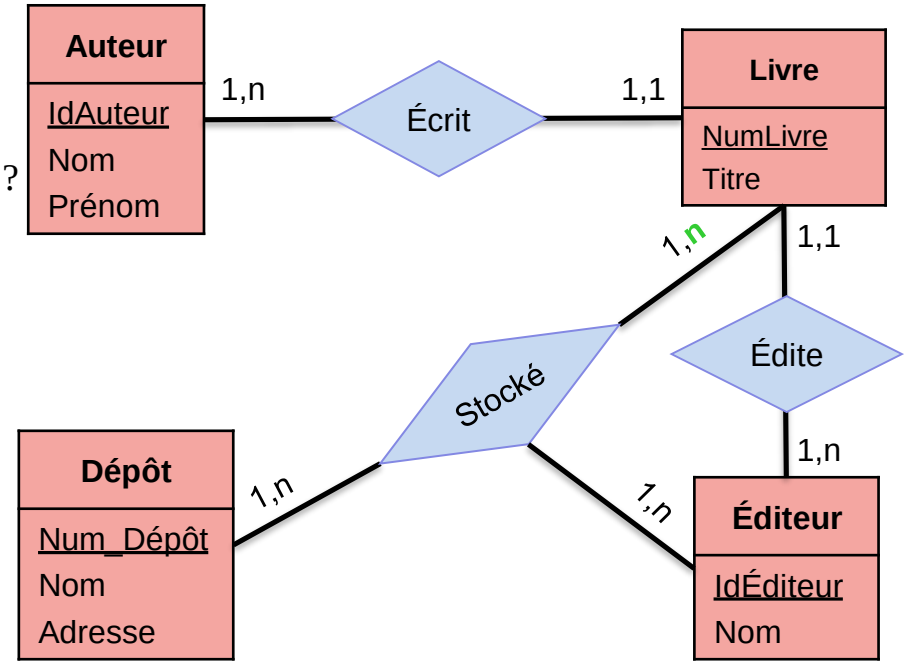
Réponse :

Non, chaque éditeur a publié au moins un livre, car sa cardinalité minimale prend la valeur 1.

‘ Pratique ’

Ex-2 :

À partir de ce diagramme, répondre aux questions suivantes :



5. Est-il possible de stocker un livre dans plusieurs dépôts ?

Réponse :

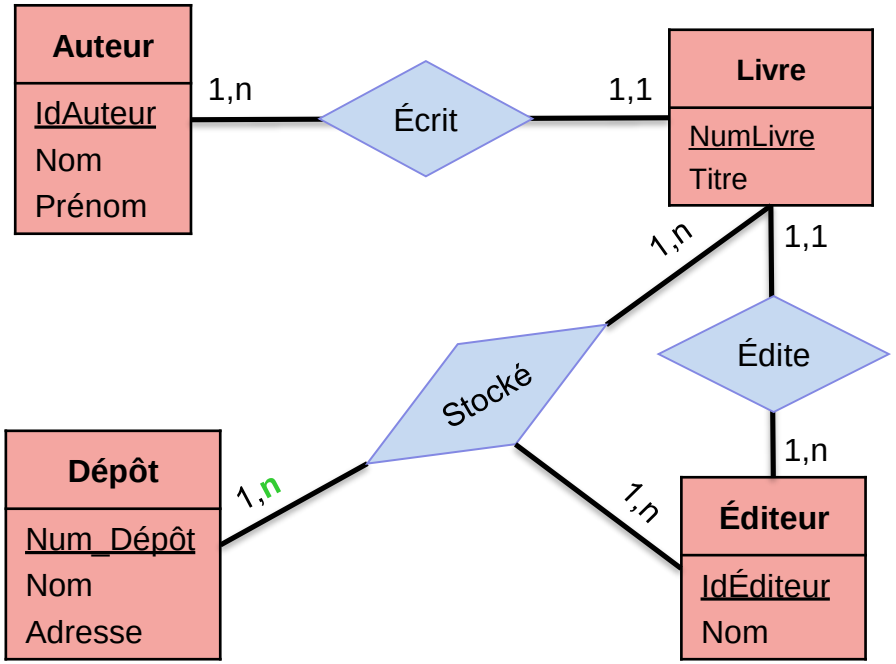
Oui, il peut le faire car sa cardinalité maximale vaut n.

‘ Pratique ’

Ex-2 :

À partir de ce diagramme, répondre aux questions suivantes :

6. Est-il possible qu’un dépôt peut être utilisé par de nombreux éditeurs ?



Réponse :

Oui, un dépôt peut être utilisé par de nombreux éditeurs, car sa cardinalité maximale vaut n.

‘Normalisation’

La **normalisation** est un algorithme qui consiste de partir d'une **table universelle** composée de la totalité des attributs pour avoir **plusieurs tables**.

Cet algorithme de normalisation est constitué des étapes qui vérifient si les tables sont dans états **bien définies**. Ces états s'appellent : **les formes normales**.

Formes normales :

- ✓ Les formes normales s’appliquent aux **entités** et aux **associations**.
- ✓ Elles ont pour objectif de vérifier la **non redondance** de l’information dans le **modèle** et de proposer les **transformations** applicables sans **perte** d’informations.

‘Normalisation’

Une **relation** est en **1ère forme normale** si :

- ✓ Elle possède au moins une clé ;
- ✓ Tous ses attributs sont atomiques : ils ne sont pas des propriétés répétitives ou décomposables.

Exemple :

Etudiant
Nom
Prénom
Age
Université

✗ Clé

Tableau correspondant

✗ Atomiques

La table Etudiant n’est pas en **1ère forme normale**

Nom	Prénom	Age	Université
HASANI	Khadija	18	Moulay Ismaîl
RAHIMI	Rime	21	Moulay Ismaîl
FATHI	Khadija	19	Moulay Ismaîl
RAYES	Achraf	20	Mohamed V
FATHI	Khadija	19	Mohamed V

‘Normalisation’

Processus de mise en **1ère forme normale** :

- ✓ Si la relation ne possède aucune clé, ajouter une ;
- ✓ Sortir les attributs non atomiques et les transformer en nouvelle table.

Exemple :

Etudiant
Nom
Prénom
Age
Université

1ère forme normale

Etudiant
CNE
Nom
Prénom
Age

+

Université
IdUniv
Nom
Adresse

‘ Normalisation ’

On dit qu'il y a Dépendance Fonctionnelle (DF) entre un attribut A et un autre B, si A détermine B.

Une **relation** est en **2ème forme normale** :

- ✓ La table soit déjà en 1ère forme normale ;
- ✓ La table possède une clé élémentaire (formée à partir d'un seul attribut) ;
- ✓ Si la table possède une clé composée, les autres attributs doit dépendre de la totalité de cette clé.

Exemple :

✓2FN

Personne
CIN
Nom
Prénom
Age

✓2FN

Cours
Titre
Filière
Horaire
Local

✗2FN

Employé
Nom
IdProfession
NumBureau
Salaire

✗2FN

Location
IdClient
IdAppartement
Montant
AdresseAppartement

‘Normalisation’

Processus de mise en **2ème forme normale** :

- ✓ Conserver dans la table initiale les attributs dépendants de la **totalité** de la clé.
- ✓ Regrouper dans une nouvelle table les champs dépendants d'une partie de la clé, et faire cette partie la clé primaire de la nouvelle table.

Exemple :

Location
IdClient
IdAppartement
Montant
AdresseAppartement

2ème forme normale

Location
IdClient
IdAppartement
Montant

+

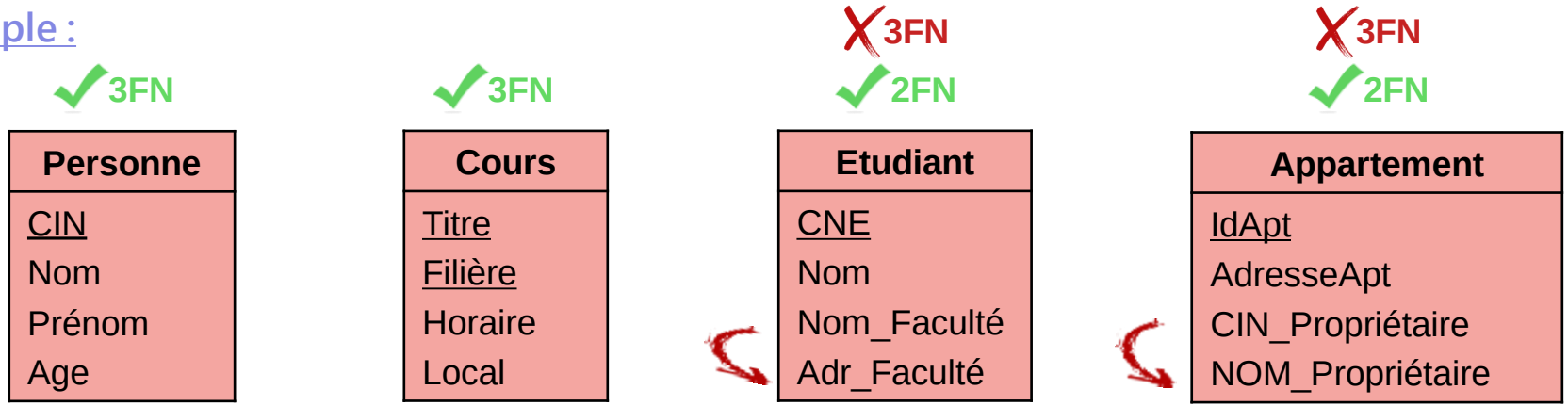
Appartement
IdAppartement
AdresseAppartement

‘Normalisation’

Une **relation** est en **3ème forme normale** :

- ✓ La table soit déjà en 2ème forme normale ;
- ✓ Tous les attributs dépendent directement de la clé et pas d’autres attributs (pas de transitivité).

Exemple :



‘Normalisation’

Processus de mise en **3ème forme normale** :

- ✓ Conserver dans la table initiale les attributs dépendants **directement** de la clé.
- ✓ Regrouper dans une nouvelle table les attributs dépendants transitivement de la clé. L’attribut de transition reste **dupliqué** dans la table initiale, et devient la **clé primaire** de la nouvelle table.

Exemple :

Appartement
IdApt
AdresseApt
CIN_Propriétaire
NOM_Propriétaire

3ème forme normale

Appartement
IdApt
AdresseApt
CIN_Propriétaire

+

Propriétaire
CIN_Propriétaire
NOM_Propriétaire

‘Normalisation’

Une **relation** est en **forme normale de Boyce-Codd (BCFN)** :

- ✓ La table soit déjà en 3ème forme normale ;
- ✓ Tout attribut qui appartient à la clé ne dépend pas à l’un des autres attributs.

Exemple :

✓BCFN

Personne
CIN
Nom
Prénom
Age

✓BCFN

Cours
Titre
Filière
Horaire
Local

✗BCFN
✓3FN

Personne
CIN
Région
Nom
Ville

✗BCFN
✓3FN

Faculté
Nom_Faculté
Nom_Dept
Adresse
Nom_Diplôme

‘ Normalisation ’

Processus de mise en **forme normale de Boyce-Codd (BCFN)** :

- ✓ Conserver dans la table initiale tout attribut n'est pas source d'une Dépendance Fonctionnelle (DF) vers une partie de la clé.
- ✓ Remplacer dans la table initiale la partie de la clé par son attribut source d'une DF.
- ✓ Regrouper dans une nouvelle table la partie de la clé et son attribut source d'une DF, et faire cette dernière la clé primaire de la nouvelle table.

Exemple :

Faculté
Nom_Faculté
Nom_Dept
Adresse
Nom_Diplôme

BCFN ⇒

Faculté
Nom_Faculté
Nom_Diplôme
Adresse

+

Diplôme
Nom_Diplôme
Nom_Dept

‘Pratique’

Soit le schéma relationnel donné ci-dessous d’une base de données pour une établissement universitaire :

Département (CNE, Note, Filière, Cours, Module, Num_Prof, NomProf, Nom_Etud, Nb-h)

1. Déterminer les dépendances fonctionnelles possibles ?

Réponse :

1. Les dépendances fonctionnelles :

- CNE → Nom-Etud, Filière
- Num-Prof → Nom-Prof
- Cours → Module
- Cours, Module → Nb-h
- Filière, Cours, Module → Num_Prof, Nom-Prof
- CNE, Cours, Module → Note

‘ Pratique ,

Soit le schéma relationnel donné ci-dessous d’une base de données pour une établissement universitaire :

Département (CNE, Note, Filière, Cours, Module, Num_Prof, Nom-Prof, Nom_Etud, N-h)

2. Quelle est la clé de cette relation ?

Réponse :

1. Les dépendances fonctionnelles :
   - CNE → Nom_Etud, Filière
   - Num_Prof → Nom-Prof
   - Cours → Module
   - Cours, Module → N-h
   - Filière, Cours, Module → Num_Prof, Nom-Prof
   - CNE, Cours, Module → Note

2. La clé :
   - ✓ CNE
   - ✓ Cours

‘ Pratique ’

3. Mettre cette relation en 3FN ?

Réponse :

Département (CNE, Cours, Note, Filière, Module, Num_Prof, NomProf, Nom_Etud, Nb-h)

✓ 1FN ✗ 2FN

Clé

Attributs Atomiques

- CNE → Nom-Etud, Filière
- Cours → Module, Nb-h
- Num-Prof → Nom-Prof
- CNE, Cours → Note

✓ 3FN ▪ **Etudiants** (CNE, Nom_Etud, Filière)

✓ 3FN ▪ **Matières** (Cours, Module, Nb-h)

✗ 3FN ▪ **Département** (CNE, Cours, Note, Num_Prof, NomProf)

✓ 3FN ▪ **Profs** (Num_Prof, NomProf)

✓ 3FN ▪ **Département** (CNE, Cours, Note)

Lorsque le **nom** de la relation n’est pas **significatif**, on peut le renommer.

‘ Pratique ’

3. Mettre cette relation en 3FN ?

Réponse :

Département (CNE, Cours, Note, Filière, Module, Num_Prof, NomProf, Nom_Etud, Nb-h)

✓ 1FN ✗ 2FN

Clé

Attributs Atomiques

- CNE → Nom-Etud, Filière
- Cours → Module, Nb-h
- Num-Prof → Nom-Prof
- CNE, Cours → Note

✓ 3FN ▪ **Etudiants** (CNE, Nom_Etud, Filière)

✓ 3FN ▪ **Matières** (Cours, Module, Nb-h)

✓ 3FN **Profs** (Num_Prof, NomProf)

✓ 3FN **Examens** (CNE, Cours, Note)

Modèle Relationnel

03

Passage du E/A au MRD, L'algèbre relationnelle.

Schéma de Relation

Règles de passage du E/A au MLR

Règle numéro 1 :

a) Chaque entité d'un modèle **E/A** devient une **relation**, c'est à dire une **table**.

b) Son **identifiant** devient la **clé primaire** de la relation.

c) Les autres **propriétés** deviennent les **attributs** de la relation.

Exemple :

Entité

Personne
CIN Nom Prénom

Relation

Personne (CIN, Nom, Prénom, Age)

SGBD

CIN	**Nom**	**Prénom**
MA345	NAJI	Kaouter
…		

Schéma de Relation

Règles de passage du E/A au MLR

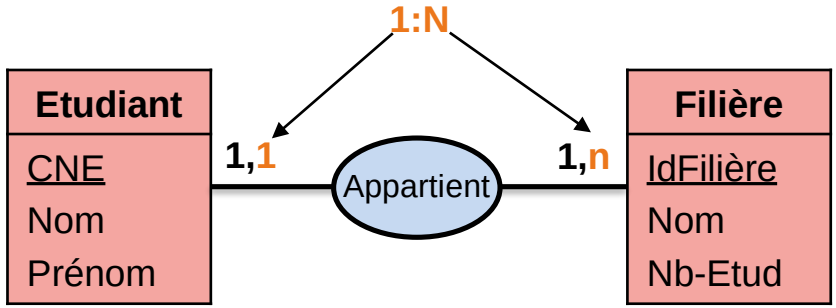
Règle numéro 2 :

Une association de type **1:N** :

a) se traduit par la création d'une **clé étrangère** dans la relation correspondante à l'entité côté « **1** ».

b) cette clé étrangère référence la **clé primaire** de la relation correspondant à l'entité côté « **N** ».

Exemple :



La clé étrangère est précédée d'un #

Schémas Relationnels

✓ **Etudiant** (CNE, Nom, Prénom, #IdFilière)

✓ **Filière** (IdFilière, Nom, Nb-Etud)

Schéma de Relation

Règles de passage du E/A au MLR

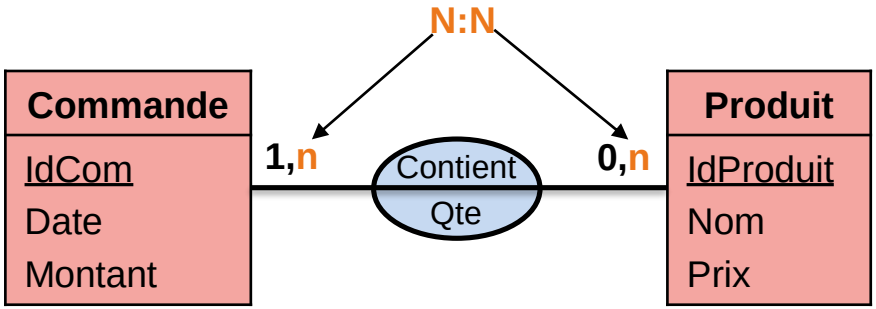
Règle numéro 3 :

Une association de type **N:N** :

a) Création d'une nouvelle table dont la **clé primaire** est l'ensemble des identifiants des entités associées.

b) Toute **propriété** de l'association devient **attribut** de la nouvelle table.

Exemple :



Schémas Relationnels

- ✓ **Commande** (IdCom, Date, Montant)
- ✓ **Produit** (IdProduit, Nom, Prix)
- ✓ **Contenir** (#IdCom, #IdProduit,Qte)

Schéma de Relation

Règles de passage du E/A au MLR

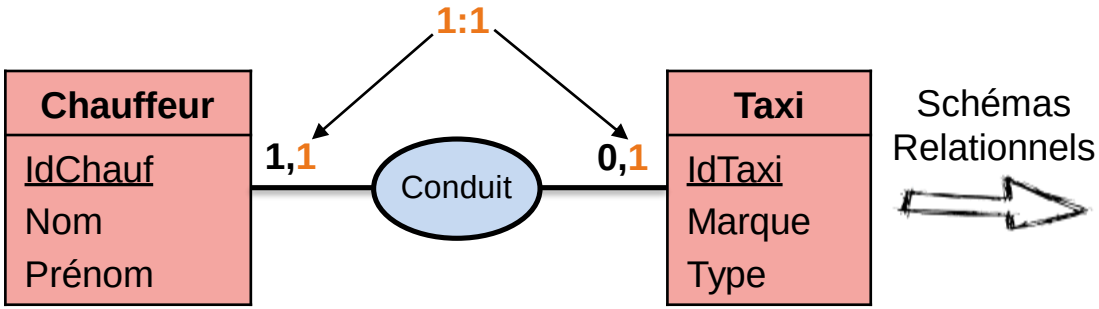
Règle numéro 4 :

Une association de type **1:1** :

a) Cela dépend fonctionnellement sur l'entité la **plus important**.

Exemple :

1 Si fonctionnellement, le **Chauffeur** est le plus important :



Schémas Relationnels

Chauffeur (IdChauf, Nom, Prénom, IdTaxi, Marque, Type)

Schéma de Relation

Règles de passage du E/A au MLR

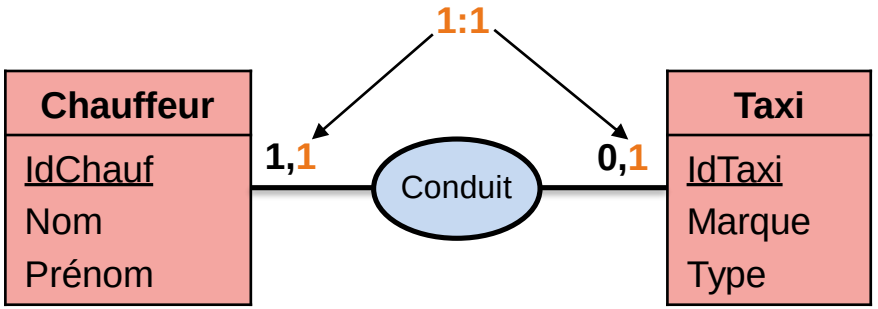
Règle numéro 4 :

Une association de type **1:1** :

a) Cela dépend fonctionnellement sur l'entité la **plus important**.

Exemple :

2 Si fonctionnellement, le **Taxi** est le plus important :



Schémas Relationnels ⇒ **Taxi** (IdTaxi, Marque, Type, IdChauf, Nom, Prénom)

Schéma de Relation

Règles de passage du E/A au MLR

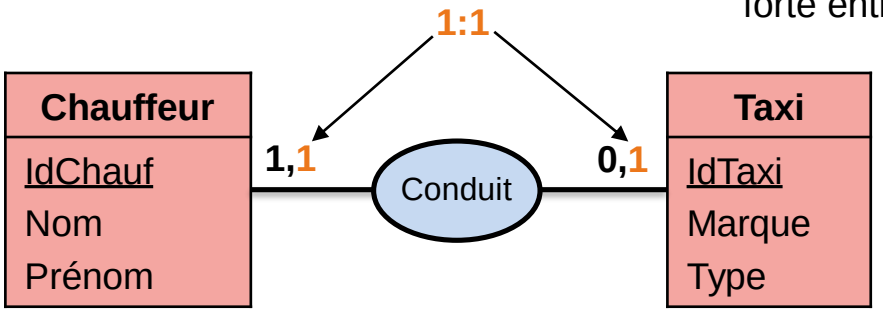
Règle numéro 4 :

Une association de type **1:1** :

a) Cela dépend fonctionnellement sur l'entité la **plus important**.

Exemple :

3 Si le modèle peut évoluer ou si on a une distinction fonctionnelle forte entre marin et voilier :



Schémas Relationnels ⇒

Chauffeur (IdChauf, Nom, Prénom, #IdTaxi)

Taxi (IdTaxi, Marque, Type)

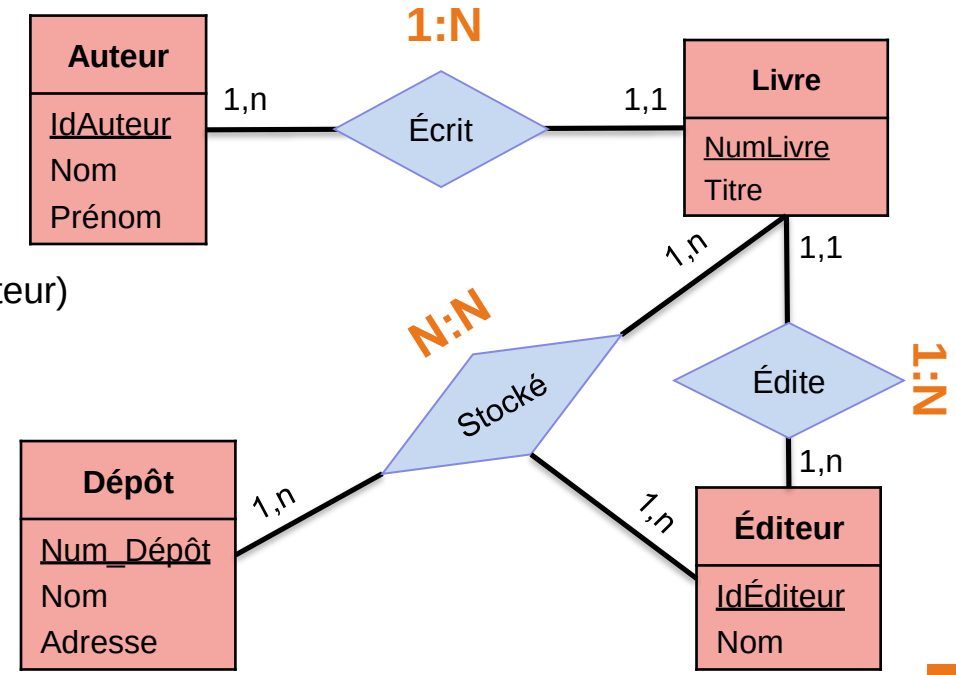
Ou

Chauffeur (IdChauf, Nom, Prénom)

Taxi (IdTaxi, Marque, Type, #IdChauf)

‘Pratique ,

Établir le modèle **MLR** (schémas relationnels) du modèle E/A suivant :



Réponse :

- ✓ **Auteur** (IdAuteur, Nom, Prénom)
- ✓ **Livre** (NumLivre, Nom, Prénom, #IdAuteur, #IdÉditeur)
- ✓ **Éditeur** (IdÉditeur, Nom)
- ✓ **Dépôt** (Num_Dépôt, Nom, Adresse)
- ✓ **Stocker** (#NumLivre, #IdÉditeur, #Num_Dépôt)

‘Algèbre relationnelle ’

Définition :

L'algèbre relationnelle est le **langage** de manipulation utilisé par le **SGBD** pour effectuer des opérations sur les relations (tables).

Remarque :

Les requêtes SQL soumises par l'utilisateur sont **traduites** par le SGBD en opérations de l'algèbre relationnelle.

Exemple :

Personnes qui ont **Samir** comme **prénom**

Personne		
CIN	**Nom**	**Prénom**
MA345	NAJI	Kaouter
FJ653	RABEH	Samir
RV481	LIMAM	Fadi
…		

Objectif :

Localiser des données dans la base qui répondent à certains **critères**.

Opérateurs ensemblistes

Les opérateurs ensemblistes sont les mêmes qu'en mathématiques, dans la théorie des **ensembles**.

Union U :

L'union de deux tables est l'ensemble des occurrences qui appartiennent soit à la première table, soit à la deuxième, soit aux deux tables. C'est la traduction du **OU** logique.

Formalisme :

R = R1 U R2

ou

R = UNION (R1 , R2)

Exemple :

R1

Nom	Prénom
NAJI	Kaouter
RABEH	Samir
LIMAM	Fadi

R2

Nom	Prénom
NAJI	Kaouter
SALEM	Rajae
LMALKI	Mounir

R

Nom	Prénom
NAJI	Kaouter
RABEH	Samir
LIMAM	Fadi
SALEM	Rajae
LMALKI	Mounir

Remarque :

Pas de duplication des n-uplets.

Opérateurs ensemblistes

Les opérateurs ensemblistes sont les mêmes qu'en mathématiques, dans la théorie des **ensembles**.

Intersection ∩ :

L'intersection de deux relations est l'ensemble des occurrences qui sont présentes dans les deux relations. C'est la traduction du **ET** logique.

Formalisme :

R = R1 ∩ R2

ou

R = INTERSECTION (R1, R2)

Exemple :

R1

Nom	Prénom
NAJI	Kaouter
RABEH	Samir
LIMAM	Fadi

R2

Nom	Prénom
NAJI	Kaouter
NAJI	Rajae
LMALKI	Mounir

R

Nom	Prénom
NAJI	Kaouter

Opérateurs ensemblistes

Les opérateurs ensemblistes sont les mêmes qu'en mathématiques, dans la théorie des **ensembles**.

Différence – :

La différence entre deux tables est l'ensemble des occurrences qui appartiennent à une table sans appartenir à la seconde. **Attention**, cette opération a un sens.

Formalisme :

R = R1 - R2

ou

R = DIFFERENCE (R1, R2)

Exemple :

R1	Nom	Prénom
	NAJI	Kaouter
	RABEH	Samir
	LIMAM	Fadi

R2	Nom	Prénom
	NAJI	Kaouter
	NAJI	Rajae
	LIMAM	Fadi

R1-R2	Nom	Prénom
	RABEH	Samir

R2-R1	Nom	Prénom
	NAJI	Rajae

Remarque :

L'opération différence est **non commutative.**

Opérateurs ensemblistes

Les opérateurs ensemblistes sont les mêmes qu'en mathématiques, dans la théorie des **ensembles**.

Produit cartésien x :

Le produit cartésien de 2 tables consiste à combiner toutes les possibilités d'associations d'occurrences des 2 tables. Chaque ligne de R1 sera concaténée à chaque ligne de R2.

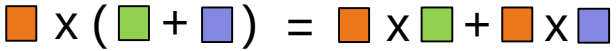
Formalisme :

R = R1 x R2

ou

R = PRODUIT (R1, R2)

Exemple :



R1

Nom	Prénom
NAJI	Kaouter
RABEH	Samir

R2

Age	Ville
19	Er-riche
21	Rissani

R1xR2

Nom	Prénom	Age	Ville
NAJI	Kaouter	19	Er-riche
NAJI	Kaouter	21	Rissani
RABEH	Samir	19	Er-riche
RABEH	Samir	21	Rissani

'Opérateurs relationnels ,

Les opérateurs **relationnels** sont spécifiques à l'algèbre relationnelle.

Sélection σ :

La sélection consiste à extraire d'une relation les occurrences satisfaisant aux critères de sélection.

Formalisme :

R2 = SELECTION (R1, critère(s))

ou

$R2 = \sigma_{critère(s)}(R1)$

Critères de sélection :

Opérateurs de comparaison :

- ✓ <, <=, =, >, >=, ?

Opérateurs logiques :

- ✓ **ET**, **OU** (entre deux comparaisons)
- ✓ **NON** (pour renverser la comparaison)

Exemple :

On aimerait avoir les personnes habitant à **Rissani** ?

- ✓ $\sigma_{Ville = 'Rissani'}(R1)$
- ✓ SELECTION (R1, Ville = 'Rissani')

R1

Nom	Prénom	Age	Ville
SABER	IMRANE	19	Er-riche
NAJI	Kaouter	21	Rissani
RABEH	Samir	17	Tinghir
JABRI	Yassine	25	Rissani

R2

Nom	Prénom	Age	Ville
NAJI	Kaouter	21	Rissani
JABRI	Yassine	25	Rissani

'Opérateurs relationnels ,

Les opérateurs **relationnels** sont spécifiques à l'algèbre relationnelle.

Projection Π :

La projection d'une relation consiste en la mise en place d'une nouvelle relation en ne retenant que certaines colonnes (attributs) et en supprimant les occurrences en double.

Formalisme :

R2 = PROJECTION (R1, colonne 1, colonne 2, …)

ou

R2 = Π colonne1, colonne2, … (R1)

Remarque :

- R1 est la table utilisée par la projection.
- R2 est la table résultat.
- **Pas de duplication** des occurrences.

Exemple :

R1

Nom	Prénom	Age	Ville
SABER	IMRANE	19	Er-riche
NAJI	Kaouter	21	Rissani
SABER	IMRANE	19	Er-riche
NAJI	Kaouter	21	Rissani

On aimerait avoir les **Noms** et **Ages** de toutes les personnes de **R1** ?

✓ $\Pi_{Nom, Age}$ (R1)

✓ PROJECTION (R1, Nom, Age)

R2

Nom	Age
SABER	19
NAJI	21

Opérateurs relationnels

Les opérateurs **relationnels** sont spécifiques à l'algèbre relationnelle.

Division / :

La division permet de trouver les occurrences d'une table qui sont associées à toutes les occurrences d'une autre table (qui le plus souvent est le résultat d'une sélection).

Formalisme :

R = DIVISION (R1, R2)

ou

R = R1/ R2

Remarque :

L'opération Division est **non commutative**.

Exemple :

R1

Nom	Note
SABER	14
NAJI	17
RIDANI	12
JABRI	09
NAJI	14
SABER	09

R2

Note
14
09

R1/R2

Nom
SABER

Opérateurs relationnels

Les opérateurs **relationnels** sont spécifiques à l'algèbre relationnelle.

Jointure ⋈ :

La jointure consiste à créer une nouvelle table à partir de deux tables ayant un champ commun (attribut) et vérifiant un critère de jointure.

Formalisme :

R = JOINTURE R1, R2 (**R1.attr_jointure** op_comparaison **R2.attr_jointure**)

R = R1 ⋈ R2 (**R1.attr_jointure** op_comparaison **R2.attr_jointure**)

Exemple :

R = R1 ⋈ R2 (**R1.Cours = R2.Cours**)

R1

Cours	Profs
Informatique	Naji
Biologie	Jabri
Chimie	Fadili
Géologie	Salmi

⋈

R2

Étudiant	Cours	Note
Mounir	Biologie	14
Naoual	Biologie	10
Khadija	Physique	16
Hafid	Chimie	09

⇨

R

Cours	Profs	Étudiant	Note
Biologie	Jabri	Mounir	14
Biologie	Jabri	Naoual	10
Chimie	Fadili	Hafid	09


Langage SQL
Commandes, Fonctions.
04

‘Langage SQL ,

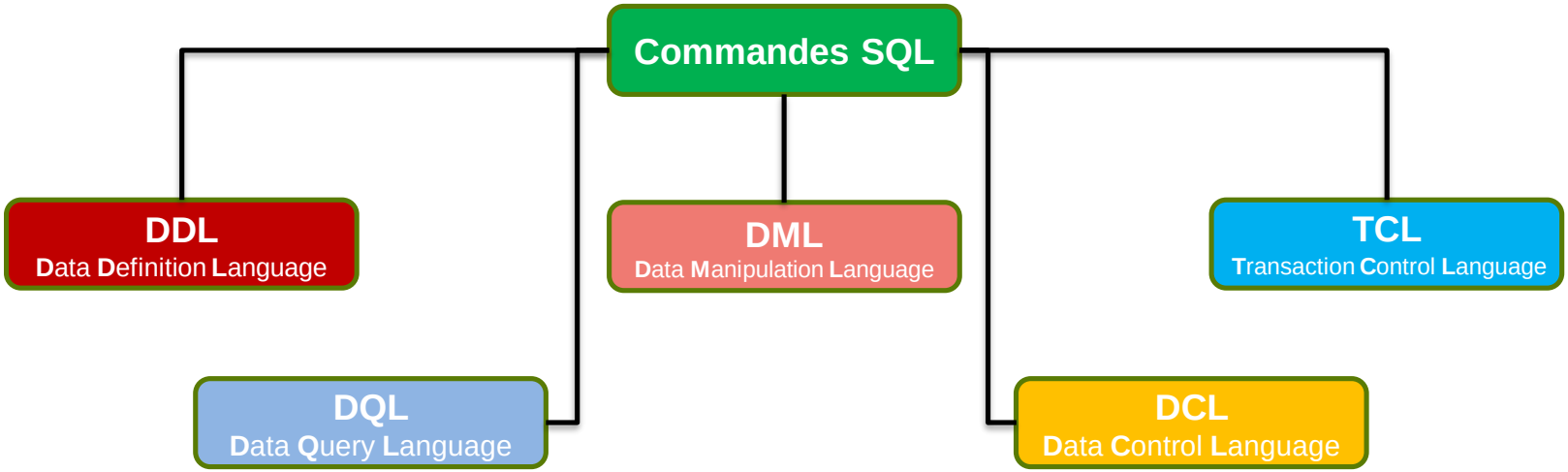
Définition :

- Qu'est ce que SQL ?
  - ✓ SQL = Structured Query Language, en français Langage de Requête Structurée.
- Un langage informatique normalisé et concret pour interagir avec le modèle relationnel :
  - ✓ Un langage de manipulation de données;
  - ✓ Un langage de description de données;
  - ✓ Un langage pour administrer la base de données et aussi gérer les contrôles d'accès.
- Fondement : L’algèbre relationnelle.
- Langage déclaratif → description du résultat escompté.

‘Commandes SQL ’

Définition :

Les commandes du langage SQL sont principalement classées en **cinq** catégories :



Commandes SQL

1. DDL :

Data **D**efinition **L**anguage : Les commandes SQL utilisées pour manipuler les **structures** de données d'une base de données, et non les données elles-mêmes.

Ex. Structures de données

Table

est un ensemble de données organisées sous forme d'un tableau.

utilisée et entretenue par le SGBD pour lui permettre de retrouver rapidement les données.

Index

User

Les comptes d'utilisateur de base de données

Commandes SQL

1. DDL :

Data **D**efinition **L**anguage : Les commandes SQL utilisées pour manipuler les **structures** de données d'une base de données, et non les données elles-mêmes.

Ex. Commandes

CREATE TABLE

→ Créer un Tableau.

Syntaxe :

```
CREATE TABLE nom_de_la_table
(
   colonne1 type_donnees,
   colonne2 type_donnees,
   colonne3 type_donnees,
   colonne4 type_donnees
)
```

Exemple :

```
CREATE TABLE Etudiant
(
   Id INT PRIMARY KEY NOT NULL,
   Nom VARCHAR(30),
   Prénom VARCHAR(30),
   Date_Naissance DATE,
)
```

Etudiant

Id	Nom	Prénom	Date_Naissance
1	Jabri	Mounir	14/06/1994
2	Kasmi	Naoual	22/01/2000
3	…	…	…

Type de données

INT : Nombre entier.

VARCHAR (Taille) : Chaînes de caractères avec une taille maximale.

PRIMARY KEY : Indiquer la clé primaire pour un index.

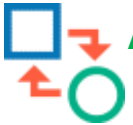
NOT NULL : Empêche d'enregistrer une valeur nulle pour une colonne.

Commandes SQL

1. DDL :

Data **D**efinition **L**anguage : Les commandes SQL utilisées pour manipuler les **structures** de données d'une base de données, et non les données elles-mêmes.

Ex. Commandes



ALTER TABLE

→ Modifier un Tableau.

Syntaxe :

```
ALTER TABLE nom_table
instruction
```

Exemple :

```
ALTER TABLE Etudiant
DROP Date_Naissance
```

Etudiant

Id	Nom	Prénom
1	Jabri	Mounir
2	Kasmi	Naoual
3	…	…

```
ALTER TABLE Etudiant
ADD Adresse VARCHAR(30)
```

Id	Nom	Prénom	Adresse
1	Jabri	Mounir	…
2	Kasmi	Naoual	…
3	…	…	…

Explication :

DROP : Supprimer une colonne du tableau.

ADD : Ajouter une colonne au tableau.

Commandes SQL

1. DDL :

Data **D**efinition **L**anguage : Les commandes SQL utilisées pour manipuler les **structures** de données d'une base de données, et non les données elles-mêmes.

Ex. Commandes

DROP TABLE

TRUNCATE TABLE

→ Supprimer/Vider un Tableau.

Exemple :

DROP TABLE Etudiant

Etudiant

Id	Nom	Prénom	Adresse
1	Jabri	…	52, Errachidia
2	Kasmi	…	102, Rissani
3	…	…	…

TRUNCATE TABLE Etudiant

Id	Nom	Prénom	Adresse
…			

Syntaxe :

ALTER TABLE nom_table
instruction

Explication :

DROP : Supprimer les données ainsi que la table qui les contient.

TRUNCATE : Vider la table. Supprimer les données sans supprimer la table en elle-même.

‘Commandes SQL ,

2. DQL :

Data **Q**uery **L**anguage : Les commandes SQL utilisées pour **récupérer** les données de la base de données.

Ex. Commandes

SELECT

→Récupération de données
dans une structure de données.

Syntaxe :

SELECT nom_champ

Sous-Commande nom_table TABLE

Sous-Commandes :

La requête **SELECT** peut posséder des sous-commandes suivantes :

- *
- **FROM** table
- **WHERE** condition
- **GROUP BY** expression
- **HAVING** condition
- { **UNION** | **INTERSECT** | **EXCEPT** }
- **ORDER BY** expression
- **LIMIT** count
- **OFFSET** start

Commandes SQL

2. DQL :

Data **Q**uery **L**anguage : Les commandes SQL utilisées pour **récupérer** les données de la base de données.

Ex. Commandes

SELECT

→Récupération de données dans une structure de données.

Syntaxe :

SELECT nom_champ

Sous-Commande nom_table TABLE

Exemple :

- Sélectionner la colonne « Nom » du tableau « Etudiant ».

SELECT Nom FROM Etudiant →

Nom
Jabri
Kasmi
…

- Sélectionner tous (*) les étudiants du tableau « Etudiant » dont (WHERE) le « Nom » est « Kasmi ».

SELECT * FROM Etudiant
WHERE Nom = 'Kasmi' →

Id	Nom	Prénom	Adresse
2	Kasmi	Naoual	102, Rissani

Commandes SQL

3. DML :

Data **M**anipulation **L**anguage : Les commandes SQL utilisées pour manipuler les **données** d'une base de données. Cette catégorie inclut la plupart des commandes SQL.

Ex. Commandes

INSERT INTO

→ Insertion de données dans une table.

Exemple :

Etudiant	Id	Nom	Prénom	Adresse
	1	Jabri	Mounir	52, Errachidia
	2	Kasmi	Naoual	102, Rissani

```
INSERT INTO Etudiant (Id, Nom, Prénom, Adresse)
VALUES ('3', 'NAJI', 'Amine', '84, Tinghir')
```

Etudiant	Id	Nom	Prénom	Adresse
	1	Jabri	Mounir	52, Errachidia
	2	Kasmi	Naoual	102, Rissani
	3	NAJI	Amine	84, Tinghir

Syntaxe :

```
INSERT INTO Table (nom_colonne_1, nom_colonne_2, ...)
VALUES ('valeur 1', 'valeur 2', ...)
```

Commandes SQL

3. DML :

Data **M**anipulation **L**anguage : Les commandes SQL utilisées pour manipuler les **données** d'une base de données. Cette catégorie inclut la plupart des commandes SQL.

Ex. Commandes

DELETE

→ Suppression de données d'une table .

Exemple :

Etudiant	Id	Nom	Prénom	Adresse
	1	Jabri	**Mounir**	52, Errachidia
	2	Kasmi	Naoual	102, Rissani

```
DELETE FROM Etudiant
WHERE Nom = 'Mounir'
```

Etudiant	Id	Nom	Prénom	Adresse
	2	Kasmi	Naoual	102, Rissani

Syntaxe :

```
DELETE FROM Table
WHERE condition
```

‘Commandes SQL ,

3. DML :

Data **M**anipulation **L**anguage : Les commandes SQL utilisées pour manipuler les **données** d'une base de données. Cette catégorie inclut la plupart des commandes SQL.

Ex. Commandes

UPDATE

→ Mettre à jour de données d'une table .

Syntaxe :

```
UPDATE Table
SET colonne_1 = 'valeur 1', ...
WHERE condition
```

Exemple :

Etudiant

Id	Nom	Prénom	Adresse
1	Jabri	Mounir	52, Errachidia
2	Kasmi	Naoual	102, Rissani

```
UPDATE Etudiant
SET Adresse = '263, Bouânane',
WHERE Id = 1
```

Etudiant

Id	Nom	Prénom	Adresse
1	Jabri	Mounir	263, Bouânane
2	Kasmi	Naoual	102, Rissani

‘Commandes SQL ,

4. DCL :

Data **C**ontrol **L**anguage : Les commandes SQL utilisées pour **contrôler l'accès** aux données d'une base de données.

Ex. Commandes

GRANT

→Autorisation d'un utilisateur à effectuer une action.

Syntaxe :

GRANT Nom_privilège
ON Structure_données
TO Utilisateur
[WITH GRANT OPTION] ;

Exemple :

- Accorder une autorisation SELECT sur la table « Etudiant » à l'utilisateur « User1 ».

  GRANT SELECT ON Etudiant TO User1

- Si on veut que l'utilisateur « User1 » peut accorder le privilège SELECT sur la table « Etudiant » à un autre utilisateur, tel que « User2 », etc. On ajout cette option :

  GRANT SELECT ON Etudiant TO User1 [WITH GRANT OPTION] ;

Remarques :

- Il y'a d'autres privilèges comme ALL, EXECUTE, INSERT, UPDATE, etc.
- Pour accorder les privilèges à tous les utilisateurs, on utilise « PUBLIC ».
- En utilisant [WITH GRANT OPTION] sur l'utilisateur «User1», si «User1» accorde un privilège à d'autre utilisateur «User2», puis, vous annulez ce privilège sur l'utilisateur «User1», l'utilisateur «User2» aura toujours ce privilège.

‘Commandes SQL ,

4. DCL :

Data **C**ontrol **L**anguage : Les commandes SQL utilisées pour **contrôler l'accès** aux données d'une base de données.

Ex. Commandes

REVOKE

→ Annulation d'une commande DCL précédente.

Exemple :

- Annuler/Supprimer le privilège SELECT sur la table « Etudiant » de l’utilisateur « User1 ».

```
REVOKE SELECT
ON Etudiant
FROM User1
```

Syntaxe :

```
REVOKE Nom_ privilege
ON Structure_données
FROM Utilisateur
```

Remarques :

- Si un utilisateur a reçu un privilège sur une structure de données de plusieurs utilisateurs, il a toujours ce privilèges sur cette structure de données jusqu'à ce que tous ceux qui ont accordé l'autorisation la révoquent.
- Vous ne pouvez pas annuler les privilèges s'ils n'ont pas été accordés initialement par vous.

‘Commandes SQL ,

5. TCL :

Transaction **C**ontrol **L**anguage : Les commandes SQL effectués sur les **transactions**. Une transaction est la plus petite unité de travail effectuée sur une base de données.

Ex. Commandes

COMMIT

→ Validation d'une transaction en cours et l’enregistrer de façon permanente.

Syntaxe :

```
COMMIT;
```

ROLLBACK

→ Annulation des modifications effectuées par une transaction depuis la dernière COMMIT ou ROLLBACK.

Syntaxe :

```
ROLLBACK;
```

SAVEPOINT

→ Enregistrement de façon temporairement une transaction afin que vous puissiez revenir à ce point chaque fois que nécessaire.

Syntaxe :

```
SAVEPOINT nom_pointsauvegarde;
```

Fonctions SQL

Les fonctions SQL permettent d'effectuer des requêtes plus élaborées, par exemple :

- **Fonctions d'agrégation :**
  - **SUM()** : calculer la somme d'un set de résultat ;
  - **MAX()** : obtenir le résultat maximum (fonctionne bien pour un entier) ;
  - **MIN()** : obtenir le résultat minimum ;
  - **COUNT()** : compter le nombre de lignes dans un résultat ;
  - **AVG()** : calculer la moyenne sur un ensemble d'enregistrement de type numérique et non nul.
- **Fonctions mathématiques / numérique**
  - **ROUND()** : arrondir la valeur et retourner soit un nombre entier, ou de choisir le nombre de chiffre après la virgule ;
  - **RAND() :** sélectionner un nombre aléatoire à virgule, compris entre 0 et 1.

Fonctions SQL

Les fonctions SQL permettent d'effectuer des requêtes plus élaborées, par exemple :

- **Fonctions de chaînes de caractères :**
  - **UPPER()** : afficher une chaîne en majuscule ;
  - **LOWER()** : afficher une chaîne en minuscule ;
  - **CONCAT()** : concaténer des chaînes de caractères ;
  - **REVERSE()** : retourner une chaîne de caractère en inversant l'ordre des caractères ;
  - **REPLACE()** : remplacer des caractères alphanumérique dans une chaîne de caractère ;
  - **LENGTH()** : calculer la longueur d'une chaîne de caractères ;
  - **SUBSTRING()/SUBSTR()** : segmenter une chaîne de caractère ;
  - **LEFT()/RIGHT()** : extraire les premiers/la fin d'une chaîne de caractères en définissant la longueur souhaité ;
  - **TRIM()** : supprimer des caractères au début et en fin d'une chaîne de caractère.

Fonctions SQL

Les fonctions SQL permettent d'effectuer des requêtes plus élaborées, par exemple :

- **Fonctions de dates et d'heures :**
  - **NOW()** : date et heure actuelle ;
  - **CURRENT_DATE()** : date actuelle ;
  - **DATE_FORMAT()** : formater une donnée DATE dans le format indiqué ;
  - **MONTH()** : extraire le numéro de mois à partir d'une date au format AAAA-MM-JJ ;
  - **DATEDIFF()** : déterminer l'intervalle entre 2 dates spécifiées ;
  - **SEC_TO_TIME()** : afficher des secondes dans un format : "HH:MM:SS" (heures, minutes et secondes).
- **Fonctions de chiffrements :**
  - **MD5()** : chiffrer une chaîne de caractère en un entier hexadécimal de 32 caractères. La fonction MD5() utilise l'algorithme **RSA**.

Fonctions SQL

Les jointures en SQL permettent d'associer plusieurs tables dans une même requête pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.

- **INNER JOIN** : jointure interne pour retourner les enregistrements quand la condition est vrai dans les 2 tables. C'est l'une des jointures les plus communes ;
- **CROSS JOIN** : jointure croisée permettant de faire le produit cartésien de 2 tables. En d'autres mots, permet de joindre chaque lignes d'une table avec chaque lignes d'une seconde table. Attention, le nombre de résultats est en général très élevé ;
- **LEFT JOIN** : jointure externe pour retourner tous les enregistrements de la table de gauche (LEFT = gauche) même si la condition n'est pas vérifié dans l'autre table ;
- **RIGHT JOIN** : jointure externe pour retourner tous les enregistrements de la table de droite (RIGHT = droite) même si la condition n'est pas vérifié dans l'autre table ;
- **FULL JOIN** : jointure externe pour retourner les résultats quand la condition est vrai dans au moins une des 2 tables ;
- **SELF JOIN** : permet d'effectuer une jointure d'une table avec elle-même comme si c'était une autre table ;
- **NATURAL JOIN** : jointure naturelle entre 2 tables s'il y a au moins une colonne qui porte le même nom entre les 2 tables SQL ;
- **UNION JOIN :** jointure d'union.

Fonctions SQL

Questions :

Facture	ID	Produit	Prix
	1	PC	3000
	2	CAMERA	2500
	3	TV	500,76

1. Calculer le montant total de la facture ?
2. Arrondir au chiffre entier le prix de l'identifiant ID = 3 ?
3. Afficher le nom de produit en minuscule de l'identifiant ID = 2 ?

Réponses :

1. **SELECT** SUM(Prix) **FROM** Facture

SUM(Prix)
6000,76

2. **SELECT** ROUND(Prix) **FROM** Facture **WHERE** ID = 3

ROUND(Prix)
501

3. **SELECT** LOWER(Produit) **FROM** Facture **WHERE** ID = 2

LOWER(Produit)
camera

Conclusion

Résumé général, Recommandations.

05

‘Résumé général ’

Les bases de données ont pris une place importante en informatique, et particulièrement dans le domaine de la gestion. L’étude des bases de données à conduit au développement de concepts, méthodes et algorithmes spécifiques, notamment pour gérer les données en support de stockage (exemple : disques durs). En effet, dès le début de la découverte de l’informatique, les informaticiens ont constaté que la taille de la mémoire vive n’est pas capable de charger toutes les données d’une base de données. Aujourd’hui, cette supposition est aussi approuvable car le masse des données augments toujours surtout après l’apparition de Web2.0 et les objets connectés. De ce fait, les bases de données futures devront être dans la mesure de gérer des grandes masses de données, distribuées géographiquement, et accessibles par des milliers d’utilisateurs.

Références

1. Merise - Guide pratique (3e édition), septembre 2018, ISBN : 9782409015342.
2. https://sql.sh/
3. http://tecfaetu.unige.ch/staf/staf-h/tassini/staf2x/Heidi/last_bd.htm

Merci

Questions ?