Université Frères Mentouri Constantine 1
Faculté Sciences de la Terre, de la Géographie et de l'Aménagement du Territoire

Système de Gestion de Base de Données

Présenté par : *M. A. BOUTELDJA*

BASE DE DONNEES

Définition 1

Un ensemble organisé d'informations avec un objectif commun.

Peu importe le support utilisé pour rassembler et stocker les données (papier, fichiers, etc.), dès lors que des données sont rassemblées et stockées d'une manière organisée dans un but spécifique, on parle de base de données.

Définition 2 (info)

Une base de données informatisée est un ensemble structuré de données enregistrées sur des supports accessibles par l'ordinateur, représentant des informations du monde réel et pouvant être interrogées et mises à jour par une communauté d'utilisateurs.

La gestion et l'accès à une base de données sont assurés par un ensemble de **programmes** qui constituent le
Système de Gestion de Base de Données **(SGBD).**

Système de Gestion de Base de Données (SGBD)

La gestion et l'accès à une base de données sont assurés par un **ensemble de programmes** qui constituent le *Système de gestion de base de données* (SGBD).

Une fois la base de données spécifiée, on peut y insérer des données, les récupérer, les modifier et les détruire. C'est ce qu'on appelle manipuler les données.

Modèle hiérarchique : lie des enregistrements dans une structure arborescente de façon à ce que chaque enregistrement n'ait qu'un seul possesseur

Modèle Réseau : possibilité d'établir des liaisons de type *n-n*, les liens entre objets peuvent exister sans restriction. Pour retrouver une donnée dans une telle modélisation, il faut connaître le chemin d'accès.

Modèle relationnel : Une base de données relationnelle est une base de données structurée suivant les principes de l'algèbre relationnelle.

Modèle relationnel/objet : La notion de *bases de données objet* ou *relationnel-objet* est plus récente et encore en phase de recherche et de développement.

Objectifs du SGBD

Indépendance physique : La façon dont les données sont définies doit être indépendante des structures de stockage utilisées.

Indépendance logique : Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données doivent être intégrées dans une vision globale.

Accès aux données : L'accès aux données se fait par l'intermédiaire d'un Langage de Manipulation de Données (LMD). Il est crucial que ce langage permette d'obtenir des réponses aux requêtes en un temps « raisonnable ». Le LMD doit donc être optimisé, minimiser le nombre d'accès disques, et tout cela de façon totalement transparente pour l'utilisateur.

Non redondance des données : Afin d'éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu'une seule fois dans la base.

Cohérence des données : Les données sont soumises à un certain nombre de contraintes d'intégrité qui définissent un état cohérent de la base. Elles doivent pouvoir être exprimées simplement et vérifiées automatiquement à chaque insertion, modification ou suppression des données. Les contraintes d'intégrité sont décrites dans le Langage de Description de Données (LDD).

Niveaux de description des données

Pour atteindre certains de ces objectifs (surtout les deux premiers), trois niveaux de description des données ont été définis par la norme ANSI/SPARC.

Le niveau externe : correspond à la perception de tout ou partie de la base par un groupe donné d'utilisateurs, indépendamment des autres. On appelle cette description le ***schéma externe*** **ou** ***vue*****.**

Le niveau conceptuel : décrit la structure de toutes les données de la base, sans se soucier de l'implémentation physique ni de la façon dont chaque groupe de travail voudra s'en servir. Dans le cas des SGBD relationnels. On appelle cette description le ***schéma conceptuel***.

Le niveau interne ou physique : s'appuie sur un système de gestion de fichiers pour définir la politique de stockage ainsi que le placement des données. Le niveau physique est donc responsable du choix de l'organisation physique des fichiers ainsi que de l'utilisation de telle ou telle méthode d'accès en fonction de la requête. On appelle cette description le ***schéma interne***.

Le Modèle Entités - Associations

Avant de modéliser un domaine sous une forme directement utilisable par un SGBD, on passe par des modélisations intermédiaires, le modèle **entités-associations** constitue l'un des premiers et des plus courants.
Ce modèle, permet une description naturelle du monde réel à partir des concepts d'entité et d'association. Ce modèle, utilisé pour la phase de conception, s'inscrit notamment dans le cadre d'une méthode plus générale : ***Merise*.**

MERISE (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise). Langage de spécification le plus répandu dans la communauté de l'informatique des systèmes d'information.
Merise propose une démarche, dite par niveaux. Les trois niveaux de représentation des données sont:

Niveau conceptuel :
(MCD) décrit les entités du monde réel, en terme d'objets, de propriétés et de relations, indépendamment de toute technique d'organisation et d'implantation des données.

Niveau logique :
(MLD) précise le modèle conceptuel par des choix organisationnels. Il s'agit d'une transcription du MCD dans un formalisme adapté à une implémentation ultérieure.

Niveau physique :
(MPD) permet d'établir la manière concrète dont le système sera mis en place (SGBD retenu).

Éléments constitutifs du modèle Entités - Associations

La représentation du modèle entités-associations s'appuie sur trois concepts de base:

Entité

Une entité *est un objet, une chose concrète ou abstraite qui peut être reconnue distinctement et qui est caractérisée par son unicité.*
Exemples d'entité : Mohamed, Omar, Livre de Math, Voiture 207 etc.

Association

Une association *(ou une relation) est un lien entre plusieurs entités.*

Exemples d'association : l'emprunt par l'étudiant Mohamed du 3 ème exemplaire du livre « Bases de Données ».

Propriété

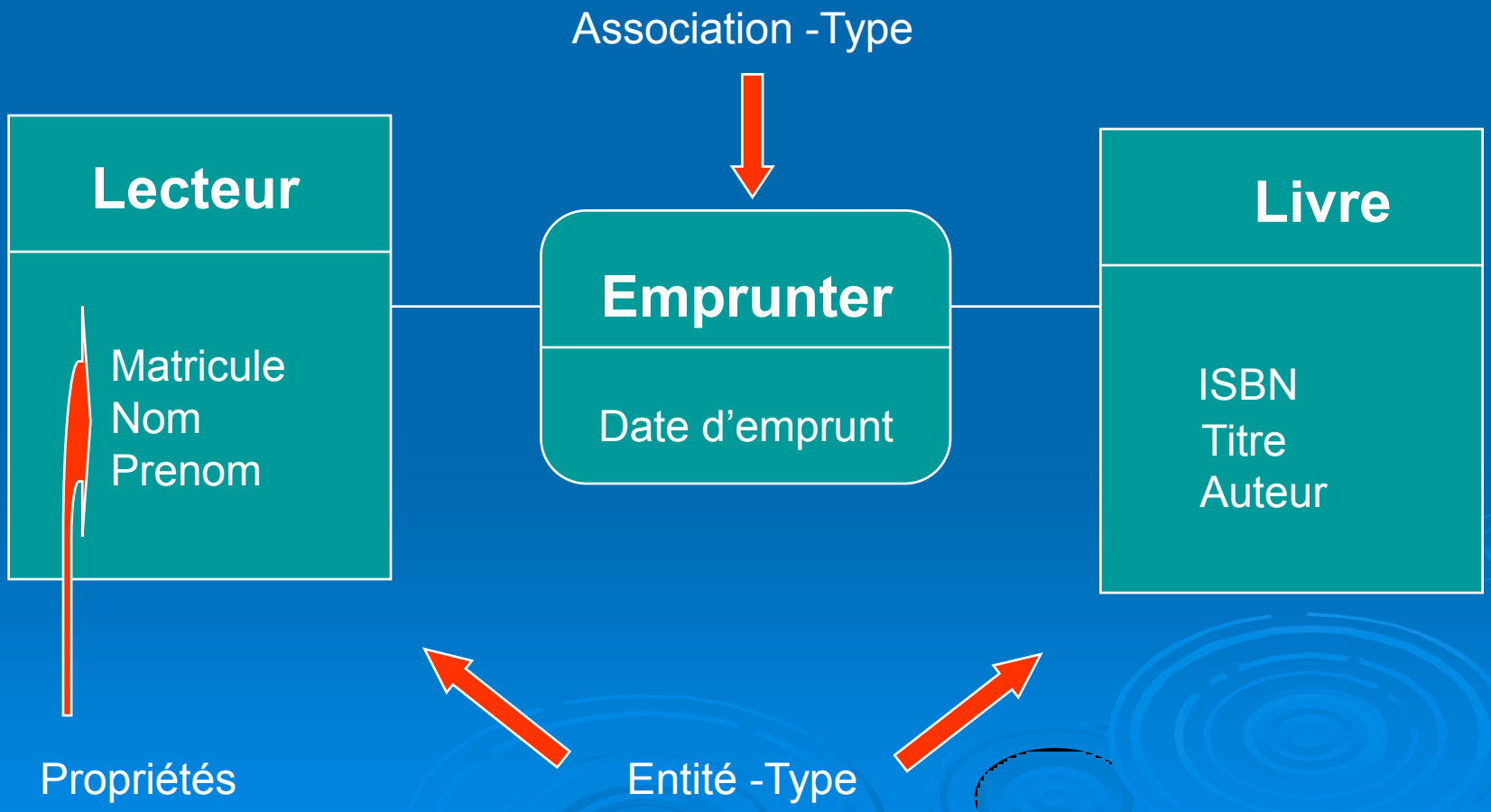
Une propriété *(ou un attribut) est une caractéristique associée à un entité-type ou à une association-type.*
Exemples d'attributs : le nom d'une personne, le titre d'un livre, la puissance d'une voiture.

Entité -Type

Association-Type

Valeur

Représentation graphique Entités - Associations



Règles

- **Un attribut ne peut en aucun cas être partagé par plusieurs entités-type ou associations-type.**
- **Un attribut est une donnée élémentaire, ce qui exclut des données calculées ou dérivées.**
- **Une entité-type et ses attributs doivent être cohérents entre eux (i.e. ne traiter que d'un seul sujet).**

Par exemple, si le modèle doit comporter des informations relatives à des parcelles et à leur propriétaires, ces informations ne doivent pas coexister au sein d'une même entité-type. Il est préférable de mettre les informations relatives aux parcelles dans une entité-type ***Parcelle*** et les informations relatives aux propriétaires dans une entité-type ***Propriétaire***.

Identifiant ou Clé

Un identifiant (ou clé) d'une entité-type ou d'une association-type est constitué par un ou plusieurs de ses attributs qui doivent avoir une valeur unique pour chaque entité ou association de ce type.

Une entité-type possède toujours une clé, et la concaténation des identifiants des entités-type liés à une association-type constitue un identifiant de cette association-type et cet identifiant n'est pas mentionné sur le modèle (il est implicite).

Définitions

Participant : *Les entités-type intervenant dans une association-type sont appelées les participants de cette association-type.*

Collection : *L'ensemble des participants d'une association-type est appelé la collection de cette association-type.*

Dimension ou Arité : *La dimension, ou l'arité d'une association-type est le nombre d' entités-type contenu dans la collection.* Et on parle d'association-type n-aire

Cardinalité : *La cardinalité d'une patte reliant une association-type et une entité-type précise le nombre de fois* ***minimal et maximal*** *d'interventions d'une entité de entité-type dans une association de association-type. La cardinalité minimale (généralement 0 ou 1) doit être inférieure ou égale à la cardinalité maximale (généralement 1 ou n).*

Les seuls cardinalités admises sont donc :

0,1 : une occurrence de entité-type peut exister tout en étant impliquée dans aucune association et peut être impliquée dans au maximum une association.

FEMME 0,1 EST MARIE A

0,n : c'est la cardinalité la plus ouverte ; une occurrence de entité-type peut exister tout en étant impliquée dans aucune association et peut être impliquée, sans limitation, dans plusieurs associations.

PERSONNE 0,n EST AUTEUR

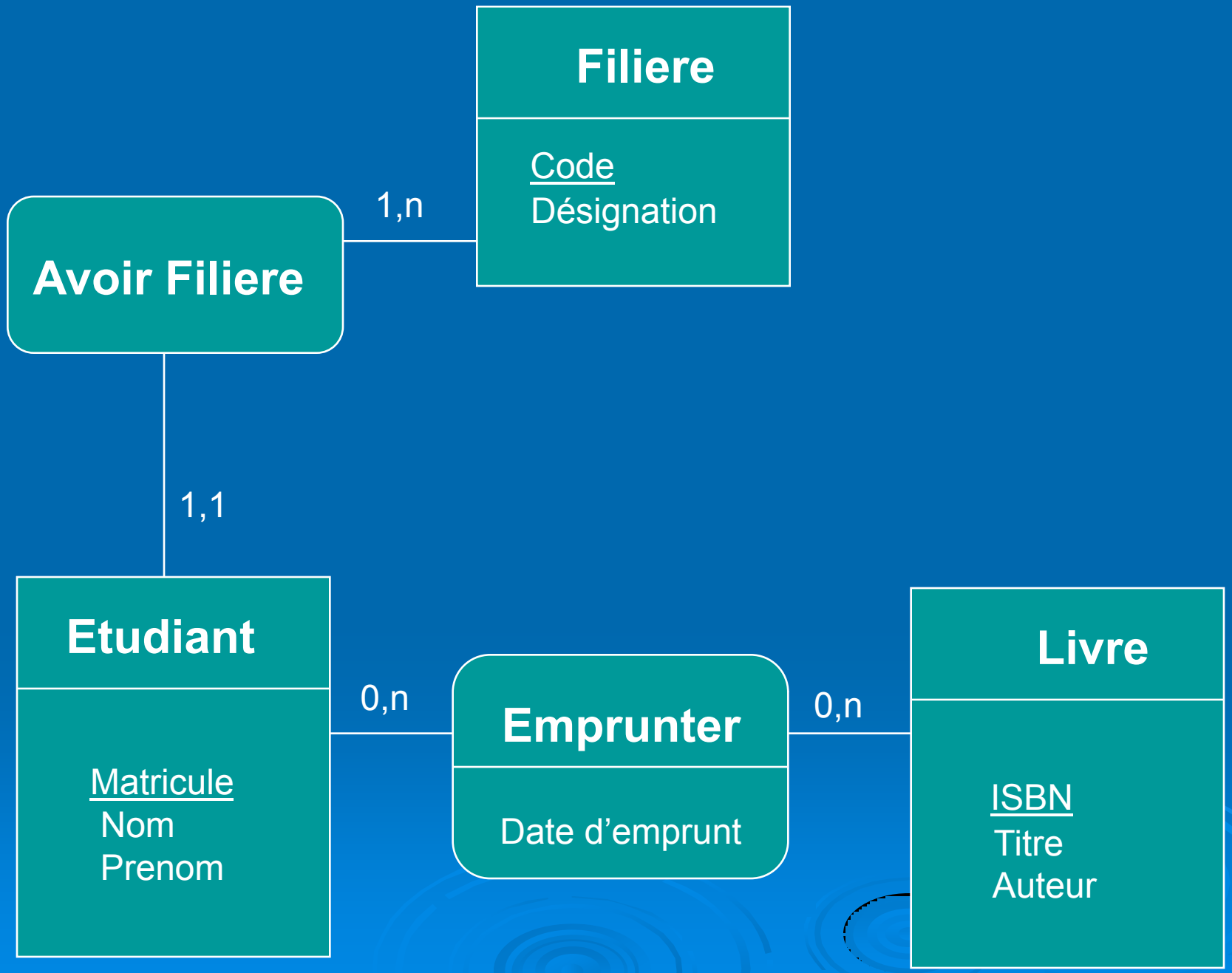
1,1 : une occurrence de entité-type ne peut exister que si elle est impliquée dans exactement (au moins et au plus) une association.

ENFANT 1,1 EST FILS

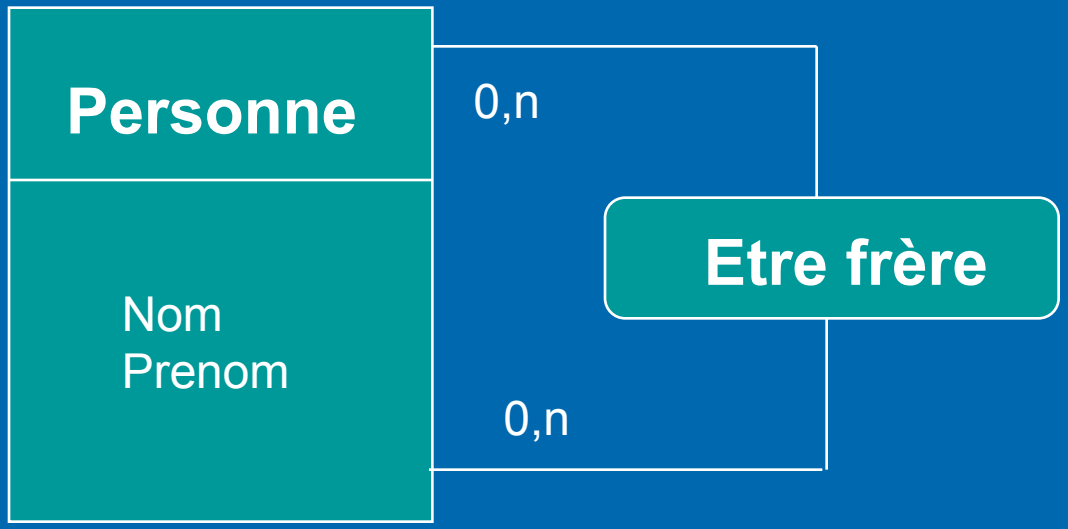
1,n : une occurrence de entité-type ne peut exister que si elle est impliquée dans au moins une association.

PROFESSEUR 1,n FAIT COURS

Les cardinalités **n-n** nécessitent généralement une décomposition


Filiere
Code
Désignation
1,n
Avoir Filiere
1,1
Etudiant
Matricule
Nom
Prenom
0,n
Emprunter
Date d'emprunt
0,n
Livre
ISBN
Titre
Auteur

Association-type réflexive : *Une association-type est qualifiée de réflexive quand elle matérialise une relation entre une entité-type et elle-même.*

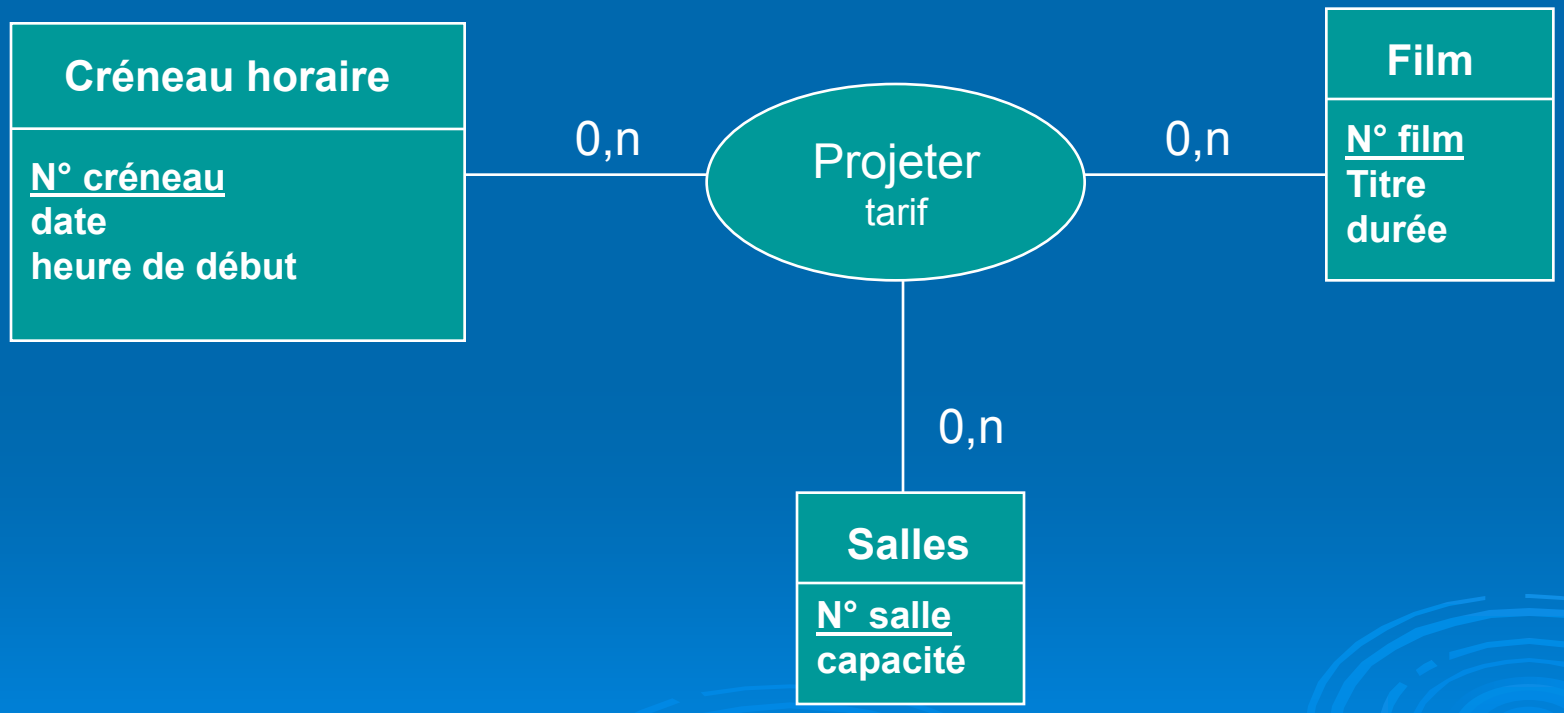


Remarque

Par abus de langage, on utilise souvent le mot entité au lieu du mot entité-type, et association au lieu de association-type, il faut cependant prendre garde à ne pas confondre les deux concepts.

Association n-aire (*n* >2)

Nous avons introduit la notion de **association *n-aire***. Ce type d'association met en relation *n* entités. Même s'il n'y a, en principe, pas de limite sur l'arité d'une association, dans la pratique on va rarement au-delà de trois. Les associations de degré supérieur à deux sont plus difficiles à manipuler et à interpréter, notamment au niveau des cardinalités.

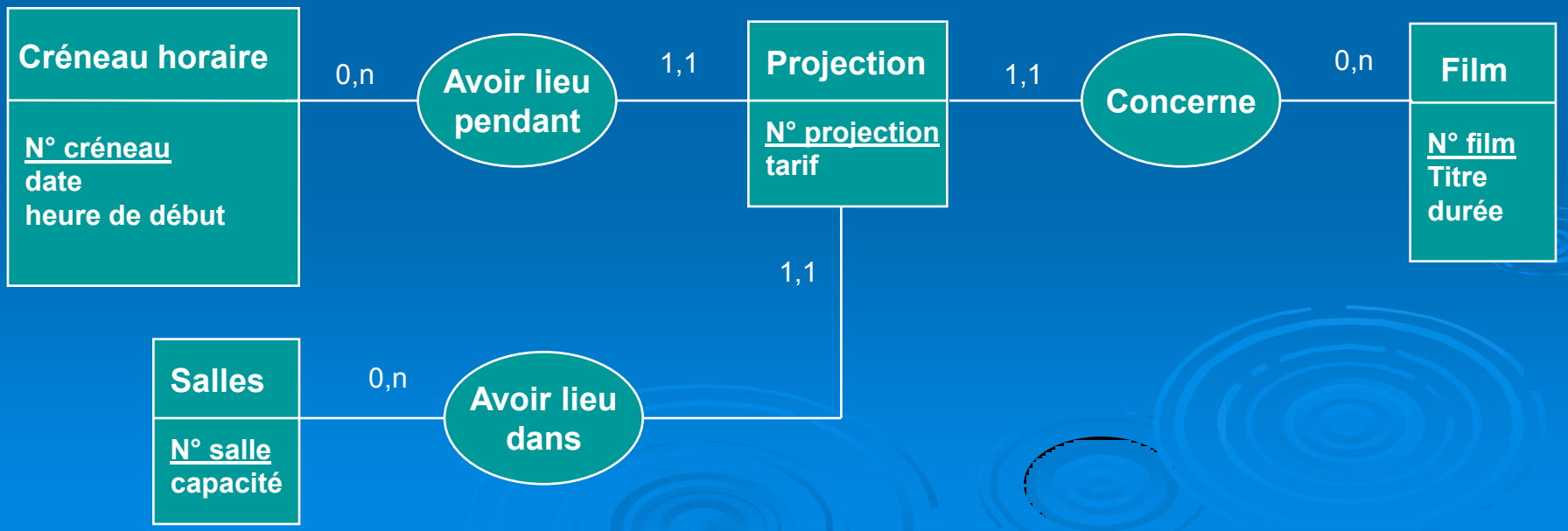


exemple d'une association ternaire entre les entités *Créneau horaire*, *Salle* et *Film*.

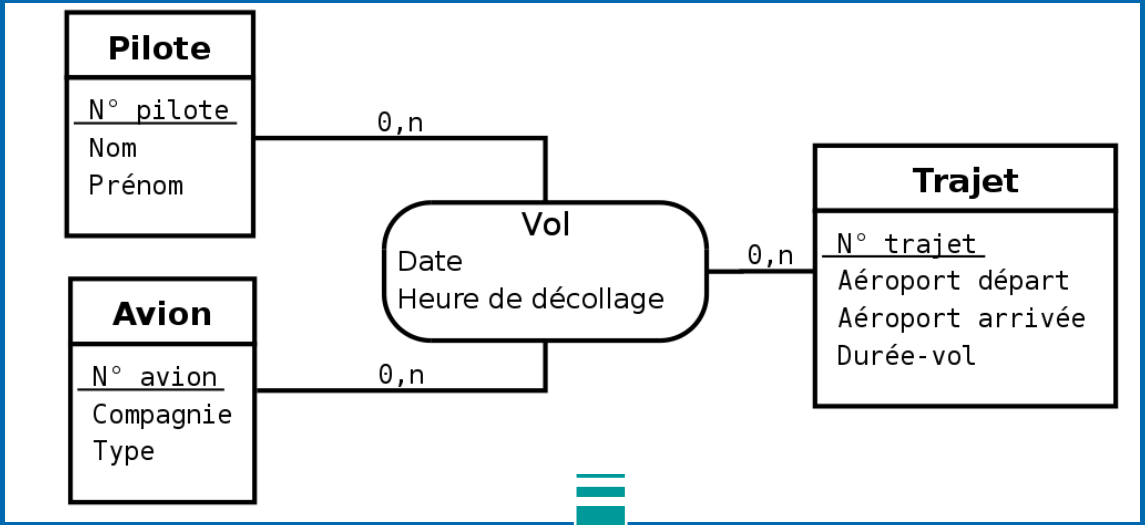
Décomposition d'une association n-aire

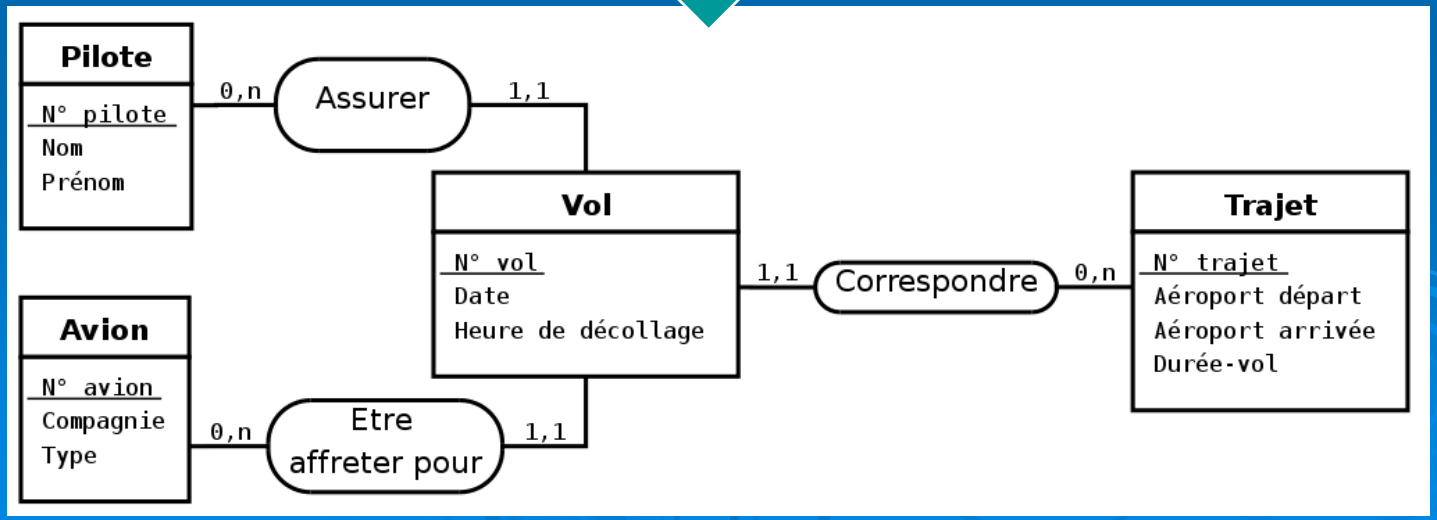
Il est toujours possible de s'affranchir d'une association n-aire (n>2) en se ramenant à des associations binaires de la manière suivante :

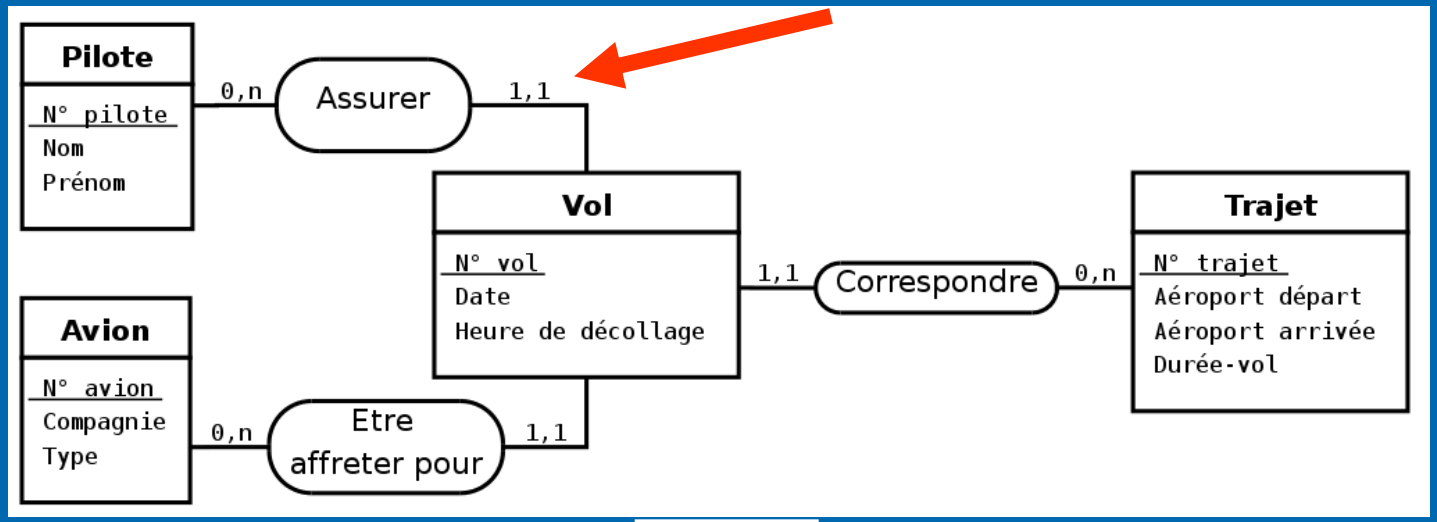
- On remplace l'association n-aire par une entité et on lui attribut un identifiant.
- On crée des associations binaire entre la nouvelle entité et toutes les entités de la collection de l'ancienne association n-aire.
- La cardinalité de chacun des associations binaires créés est 1,1 du côté de l'entité créé (celle qui remplace l'association n-aire), et 0,*n* ou 1,*n* du côté des entités de la collection de l'ancienne association n-aire.



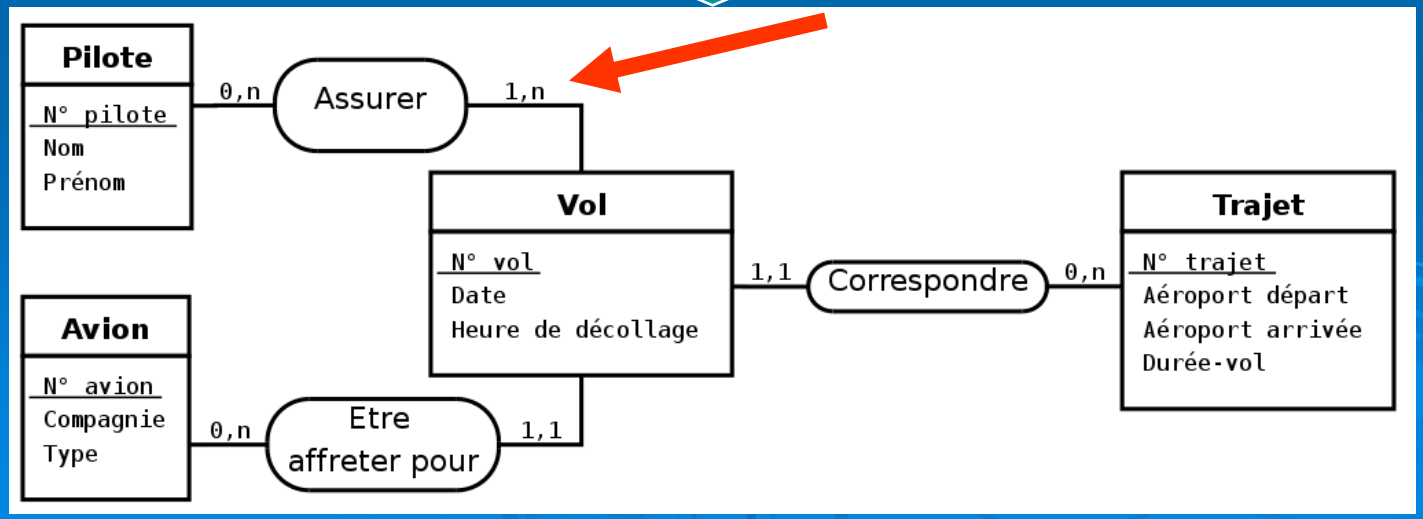
Détection d'une erreur de modélisation par décomposition d'une association n-aire







La transformation consistant à supp ssociation ternaire. Ce modèle fait immédiatement apparaître une erreu ception qui était jusque là difficile à diagnostiquer : généralement, à un v sont affectés plusieurs pilotes (par exemple le commandant de bord et) et non pas un seul.



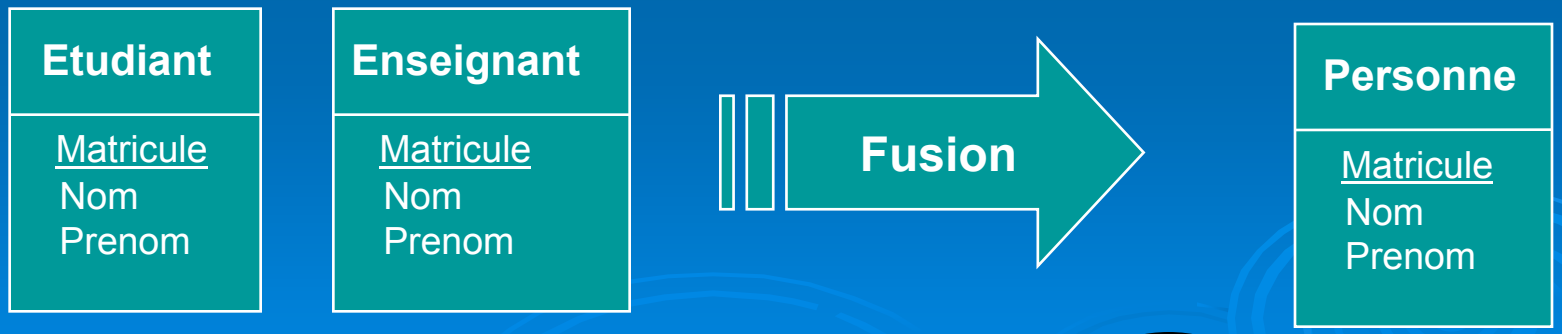
Règles de bonne formation d'un modèle Eentités-Associations

La bonne formation d'un modèle entités-associations permet d'éviter une grande partie des sources d'incohérences et de redondance. Pour être bien formé, un modèle entités-associations doit respecter certaines règles et les entités et les associations doivent être normalisées

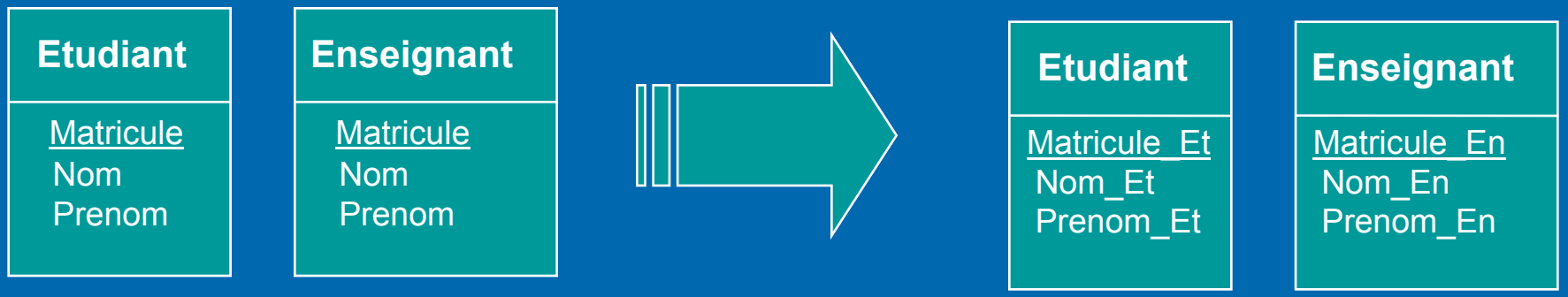
L'objectif des principes exposés dans cette section est d'aider le concepteur à obtenir un diagramme entités-associations bien formé, **ces principes ne doivent pas être interprété comme des lois**

Règles portant sur les noms

R1 : *Dans un modèle entités-associations, le nom d'une entité, d'une association ou d'un attribut doit être unique.*

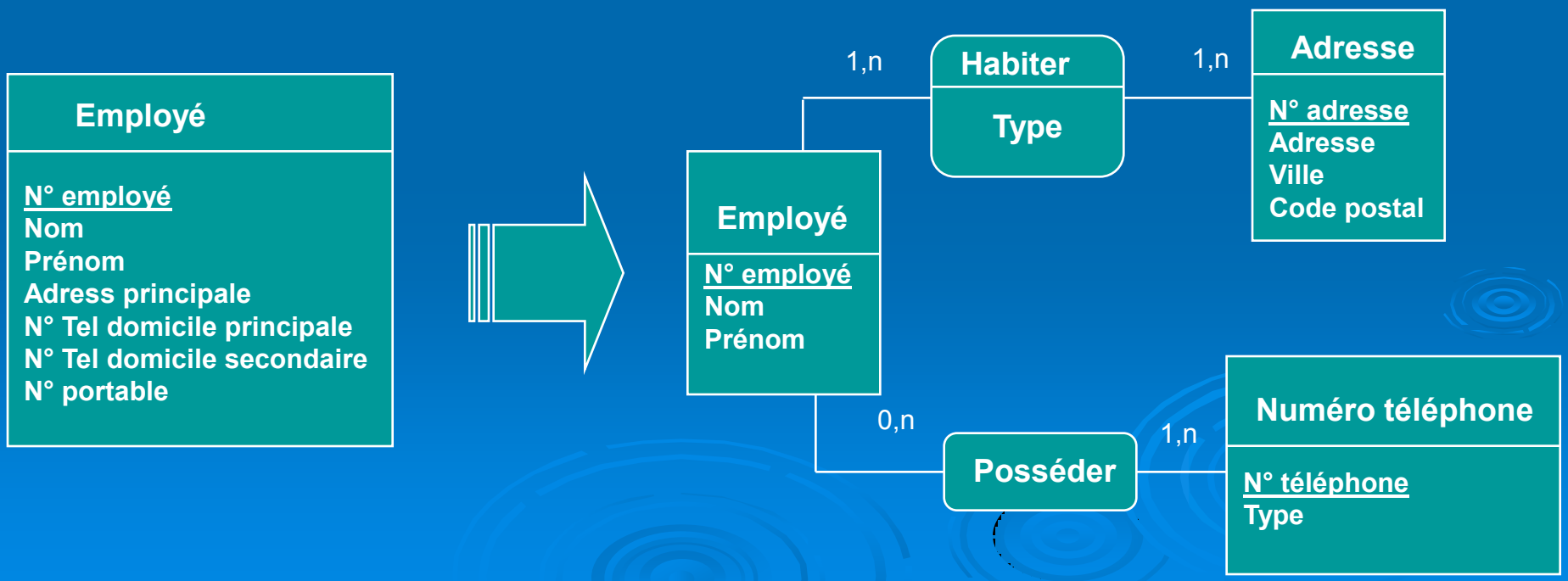


Si les deux entités doivent être séparées, on change les noms des propriétés

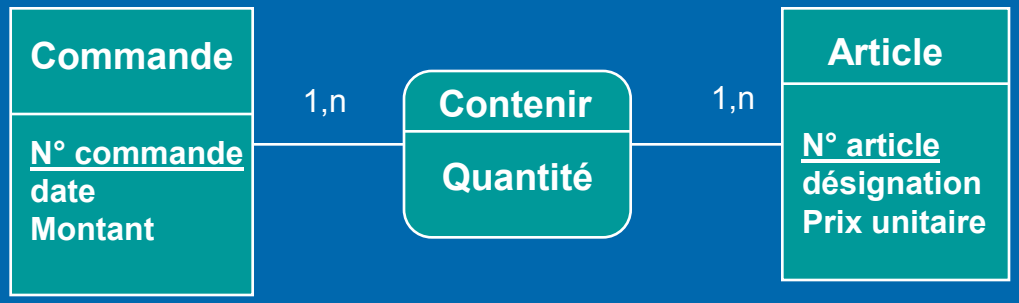


Règles de normalisation des attributs

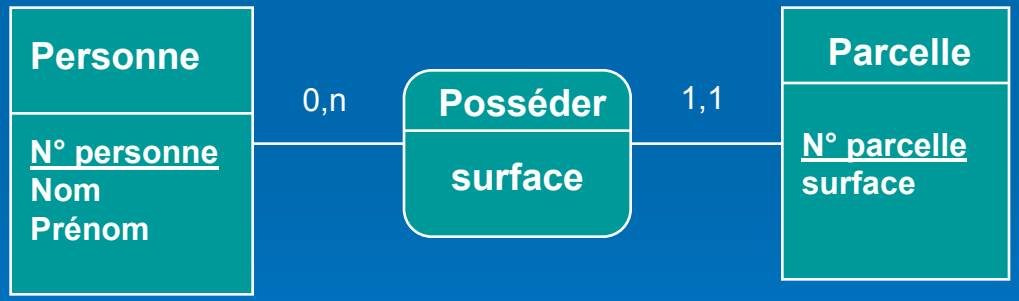
R2 : *Il faut remplacer un attribut multiple en une association et une entité supplémentaires.*



R3 : *Il ne faut jamais ajouter un attribut dérivé d'autres attributs, que ces autres attributs se trouvent dans la même entité ou pas.*

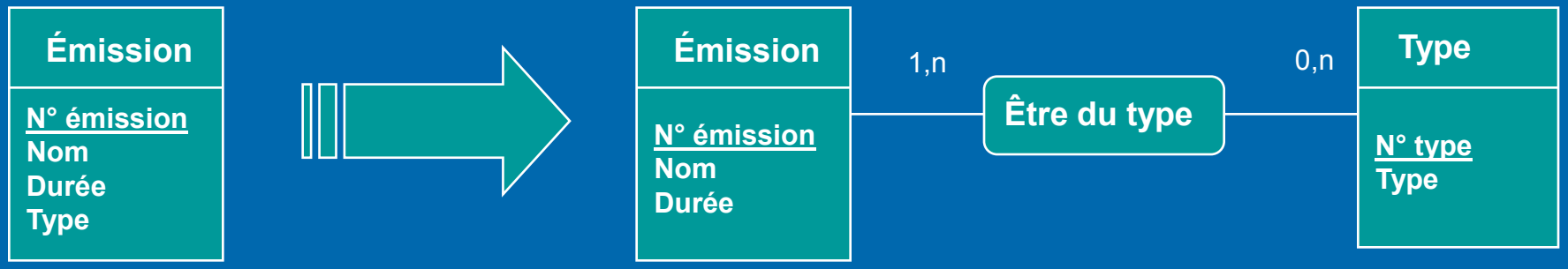


Les attributs d'une association doivent dépendre directement des **identifiants de *toutes* les entités de la collection** de l'association.



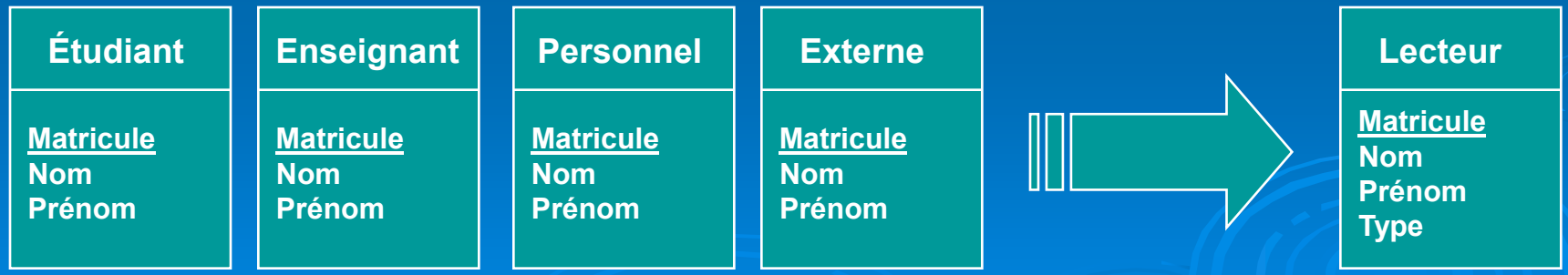
Une conséquence immédiate de cette règle est qu'un type association dont la cardinalité maximale de l'une des pattes est 1 ne peut pas posséder d'attribut

R4 : *Un attribut correspondant à un type énuméré est généralement remplacé par une entité.*

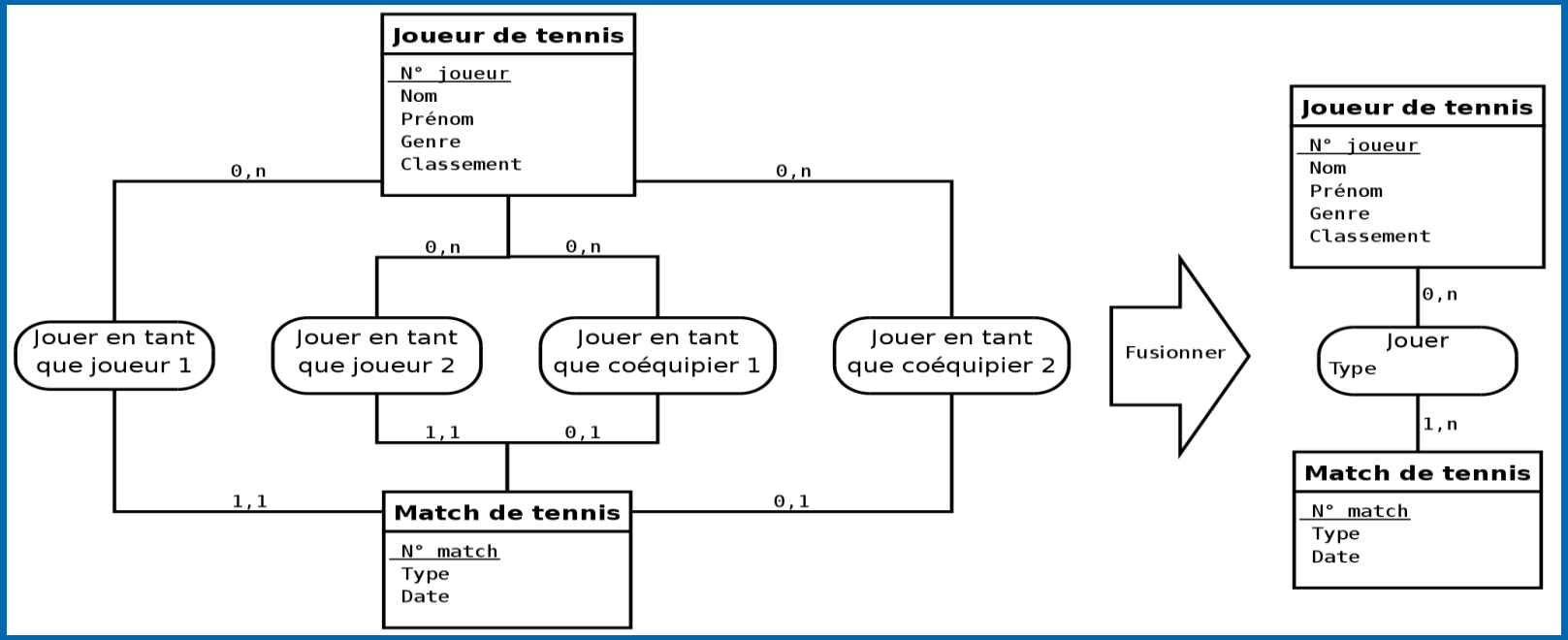


R5 : *Il faut factoriser les entités quand c'est possible.*

La spécialisation de l'entité obtenue peut se traduire par l'introduction d'un attribut supplémentaire dont l'ensemble des valeurs possibles est l'ensemble des noms des entités factorisés.



R6 : *Il faut factoriser les associations quand c'est possible*



R7 : *Lorsque les cardinalités d'une association sont toutes 1,1 c'est que l'association n'a pas lieu d'être.*

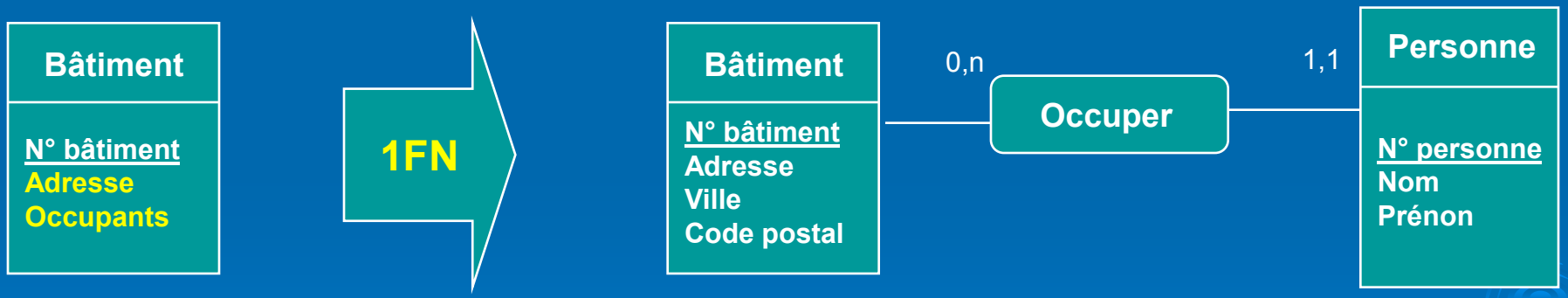
Remarque : même si toutes ses cardinalités maximale sont de 1, il est parfois préférable de ne pas supprimer l'association

Normalisation des entités et des associations

Les formes normales permettent d'éviter la redondance, source d'anomalies.
La normalisation peut être aussi bien effectuée sur un modèle entités-associations, où elle s'applique sur les entités et associations, que sur un modèle relationnel.

Première forme normale (1FN)

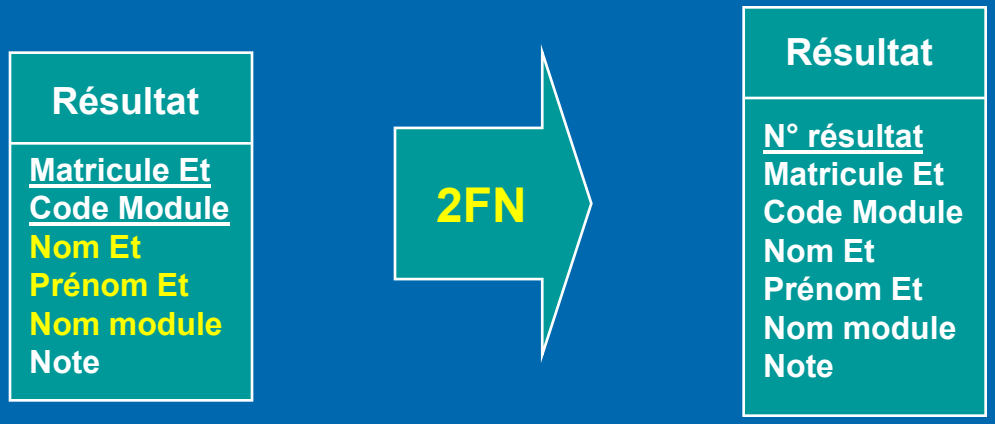
Une entité ou une association est en première forme normale si tous ses attributs sont élémentaires, c'est-à-dire non décomposables.



Un attribut composite doit être décomposés en attributs élémentaires (comme l'attribut *Adresse)* ou faire l'objet d'une entité supplémentaire (comme l'attribut *Occupants*).

Deuxième forme normale (2FN)

Une entité ou une association est en deuxième forme normale si, et seulement si, elle est en première forme normale et si tout attribut n'appartenant pas à la clé dépend de la totalité de cette clé.

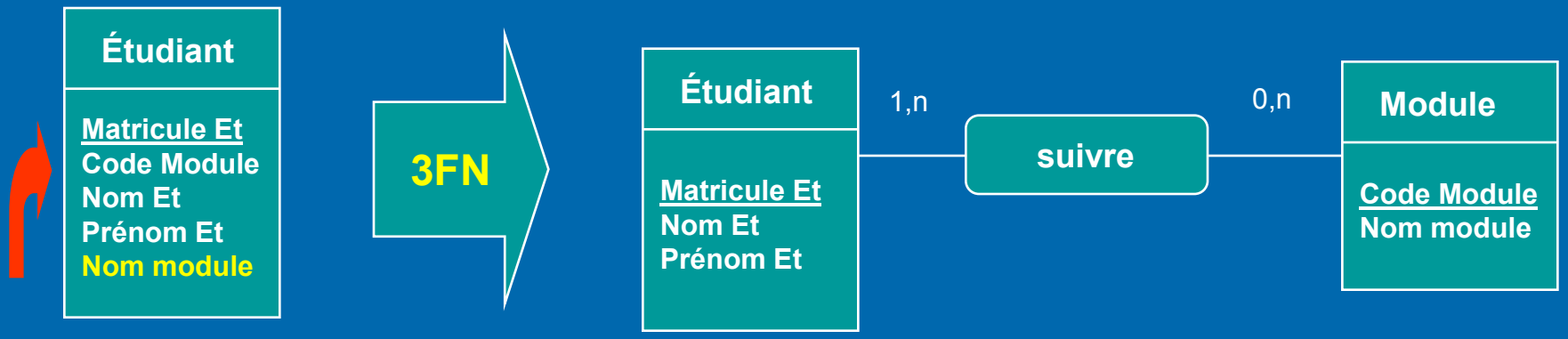


N° Résultat	Matricule Et	Nom Et	Prénom ET	Code Module	Nom Module	Note
1	12563	**RACHIDI**	**RACHID**	DB	**Base de Données**	**16**
2	12563	**RACHIDI**	**RACHID**	ALGO	Algorithmique	**12**
3	85263	KARIMI	KARIM	DB	**Base de Données**	14

Si la clé est réduite à un seul attribut, ou si elle contient tous les attributs, l'entité ou l'association est, par définition, forcément en deuxième forme normale.

Troisième forme normale (3FN)

Une entité ou une association est en troisième forme normale si, et seulement si, il est en deuxième forme normale et si tous ses attributs dépendent directement de sa clé et pas d'autres attributs.

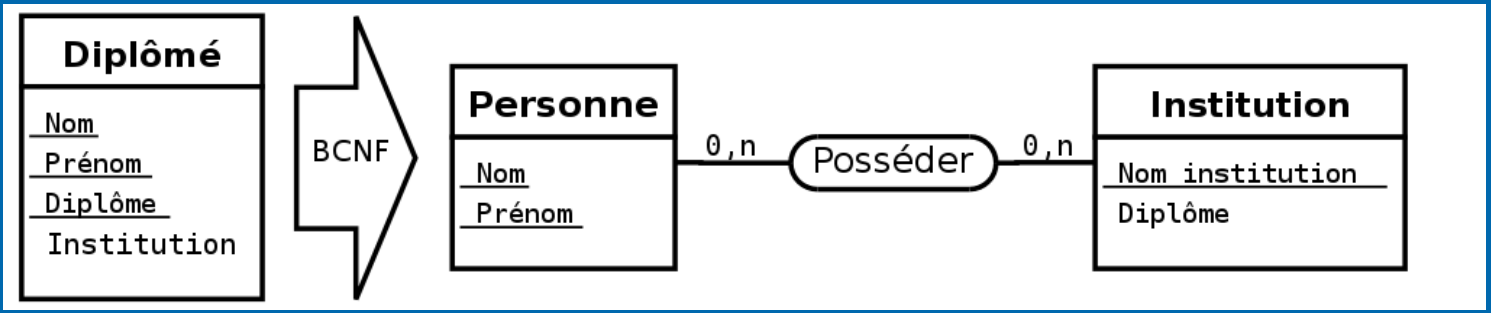


Cette normalisation peut faire apparaître des entités cachées comme le montre la figure

Une entité ou une association en deuxième forme normale avec au plus un attribut qui n'appartient pas à la clé est, par définition, forcément en troisième forme normale.

Forme normale de Boyce-Codd (BCNF)

Une entité ou une association est en forme normale de Boyce-Codd si, et seulement si, il est en troisième forme normale et si aucun attribut faisant partie de la clé dépend d'un attribut ne faisant pas partie de la clé



L'entité *Diplômé* modélise des personnes (*Nom* et *Prénom*) possédant un diplôme (*Diplôme*) d'une institution (*Institution*). On suppose qu'une même personne ne possède pas deux fois le même diplôme mais qu'elle peut posséder plusieurs diplômes différents. Une institution ne délivre qu'un type de diplôme, mais un même diplôme peut être délivré par plusieurs institutions (par exemple, plusieurs écoles d'ingénieurs délivrent des diplômes d'ingénieur). Une clé possible pour l'entité *Diplômé* est donc *Nom, Prénom, Diplôme*. L'entité obtenu est en troisième forme normale, mais une redondance subsiste car l'attribut *Institution* détermine l'attribut *Diplôme*. L'entité *Diplômé* n'est donc pas en forme normale de Boyce-Codd.

Bases de données relationnelles

Modèle Relationnel

Dans ce modèle, les données sont représentées par des tables, sans préjuger de la façon dont les informations sont stockées dans la machine. Les tables constituent donc la *structure logique* du modèle relationnel. Au niveau physique, le système est libre d'utiliser n'importe quelle technique de stockage

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

- **Les données sont organisées sous forme de tables à deux dimensions, encore appelées relations, dont les lignes sont appelées n-uplet.**
- **Les données sont manipulées par des opérateurs de l'algèbre relationnelle;**
- **L'état cohérent de la base est défini par un ensemble de contraintes d'intégrité.**

Éléments du modèle relationnel

Attribut : *Un attribut est un identificateur (un nom) décrivant une information stockée dans une base.* Exemples d'attribut : l'âge d'une personne, le nom d'une personne, le numéro de sécurité sociale.

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

Relation : *Une relation est un sous-ensemble du produit cartésien de n* ***domaines*** *d'attributs (n > 0).*
Une relation est représentée sous la forme d'un tableau à deux dimensions dans lequel **les *n* attributs correspondent aux titres des *n* colonnes**.

Schéma de relation : *Un schéma de relation précise le nom de la relation ainsi que la liste des attributs avec leurs domaines.*
Exemple : Personne(N° sécu : Entier, Nom : Chaîne, Prénom : Chaîne)

Degré : *Le degré d'une relation est son nombre d'attributs.*

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

Cardinalité : *La cardinalité d'une relation est son nombre d'occurrences*

Clé candidate : *Une clé candidate d'une relation est un ensemble minimal des attributs de la relation dont les valeurs* ***identifient à coup sûr une occurrence****.*
La valeur d'une clé candidate est donc distincte pour toutes les tuples de la relation
Toute relation a au moins une clé candidate et peut en avoir plusieurs

Clé primaire : *La clé primaire d'une relation est une de ses clés candidates. Pour signaler la clé primaire, ses attributs sont généralement* ***soulignés***

Clé étrangère : *Une clé étrangère dans une relation est formée d'un ou plusieurs attributs qui constituent une* ***clé primaire dans une autre relation***

Schéma relationnel : *Un schéma relationnel est constitué par l'ensemble des* ***schémas de relation***

Base de données relationnelle : *Une base de données relationnelle est constituée par l'ensemble des n-uplets des différentes relations du schéma relationnel*

Passage du modèle Entités-Associations au Modèle Relationnel

Règles de passage

Pour traduire un schéma du modèle Entités-Associations vers le Modèle Relationnel, on peut appliquer les règles suivantes

R1: Chaque Entité donne naissance à une Relation. Chaque propriété de cette Entité devient un attribut de la Relation. L'identifiant est conservé en tant que clé de la relation.

R2: Chaque Association dont aucune patte n'a pour cardinalité maximale 1 donne naissance à une Relation. Chaque propriété de cette Association devient un attribut de la Relation. L'identifiant, s'il est précisé, est conservé en tant que clé de la Relation, sinon cette clé est formée par la concaténation des identifiants des Entités qui interviennent dans l'Association.

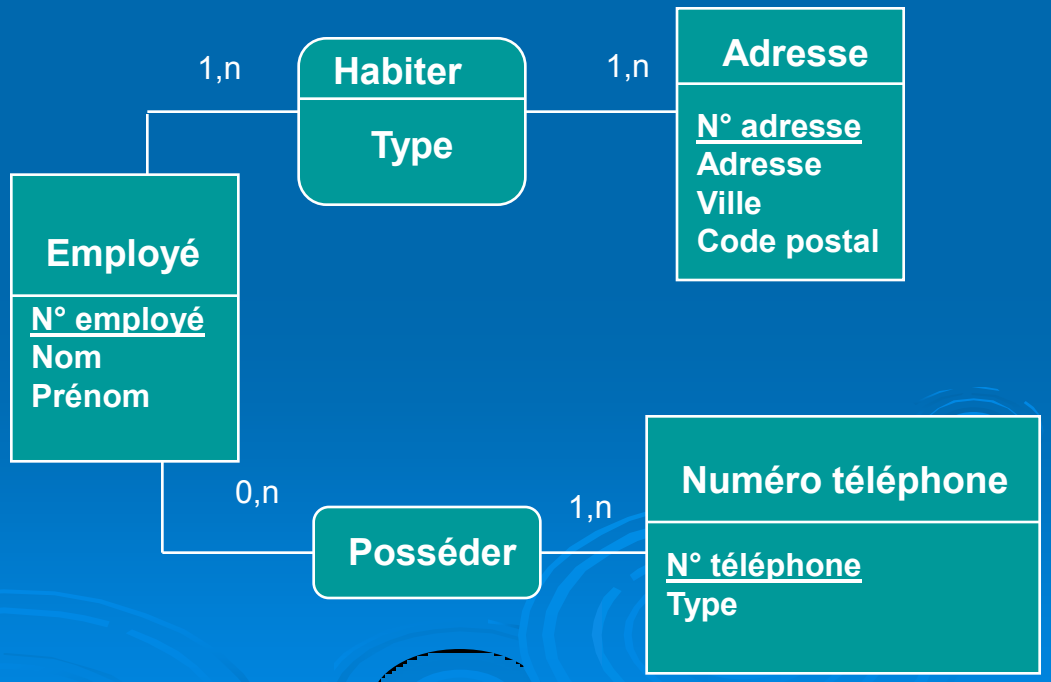
Adresse(**NumAdr**,Adresse,Ville,CodePostal)

NumTel(**NumTel**, Type)

Employé(**NumEmpl**,Nom,Prenom)

Habiter(**NumEmpl,NumAdr**,Type)

Posséder(**NumEmpl,NumTel**)



R3 : **Association un-à-un (1,1 ----- 1,1)**
L'une des entité devient un attribut de l'autre entité.

l'entité Immeuble(**N° Immeuble**, nb_étages)
l'entité Centroide(**N°Cenroide**,**coord_X, coord_Y**)

deviennent Immeuble(**N° Immeuble**, nb_étages, coord_X, coord_Y).

R4 : Association zéro-à-un (0,1 ----- 1,1)
Les deux entités restent telles quelles mais on inclut une clé externe dans la table contenant l'information sur l'entité faible (l'entité pouvant être sans occurrence c'est-à-dire celle se trouvant du côté (1,1) dans le schéma conceptuel).

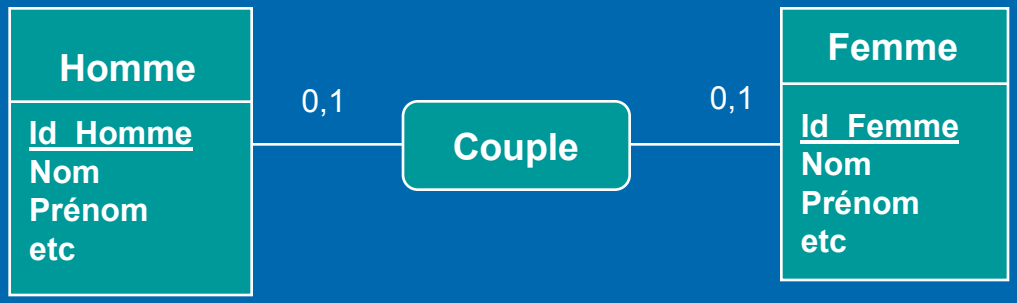
Citoyen(N°Citoyen,Nom,Prénom)
Candidat(N°Candidat,Parti)

Citoyen(N°Citoyen,Nom,Prénom)
Candidat(N°Candidat,N°Citoyen,Parti)

Le schéma relationnel adéquat

Citoyen(N°Citoyen,Nom,Prénom)
Candidat(N°Citoyen,Parti)

R5 : Association zéro-à-zéro (0,1 ---- 0,1) :
Les deux entités restent telles quelles, mais il faut en plus créer une table pour l'association.



Homme(**Id_Homme**, nom, Prénom, etc.)
Femme(**Id_Femme**, nom, Prénom, etc.)

Couple(**Id_Homme, Id_Femme**)

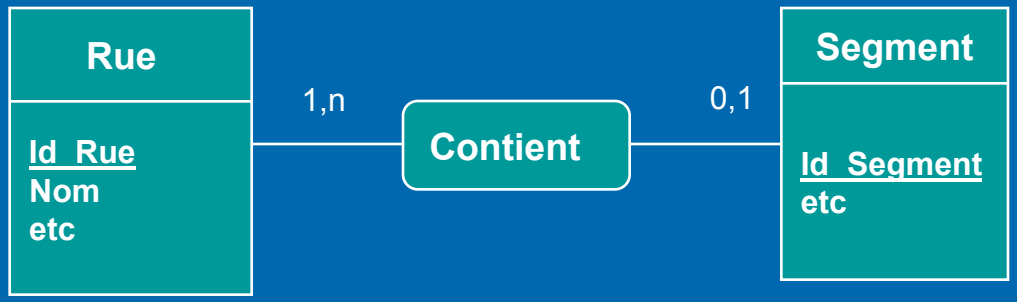
R6 : Association (1,1) ---- (1,N)

Il faut ajouter un champ (identifiant de l'autre entité) dans la table du côté "plusieurs" (1,1) qui permet d'enregistrer le lien entre les deux entités.

Commune(**Id_Commune**, nom, etc.)
Wilaya(**Id_Wilaya**, nom, etc.)

Commune(**Id_Commune**, **Id_Wilaya**,nom, etc.)
Wilaya(**Id_Wilaya**, nom, Prénom, etc.)

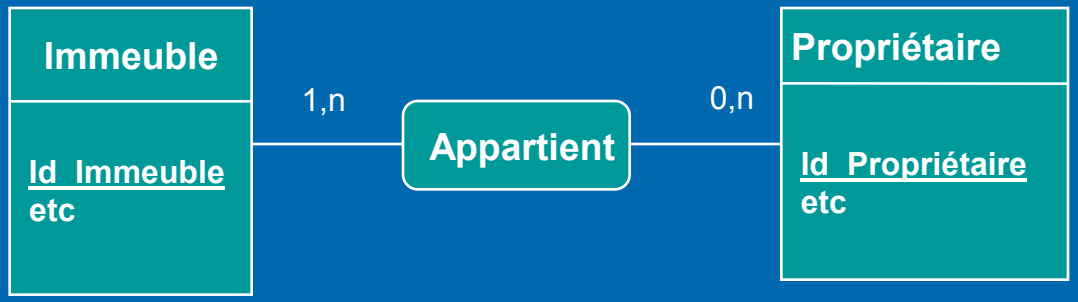
R7 : Association (0,1) ---- (1,N) ou 1,1 ----- 0,N
Il faut créer une table pour l'association en plus d'une table pour chacune des entités.



Rue(**Id_Rue**, nom, etc.)
Segment(**Id_Segment**, etc.)

Rue(**Id_Rue**, nom, etc.)
Segment(**Id_Segment**, etc.)
Rue_Segment(**Id_Rue,Id_Segment**)

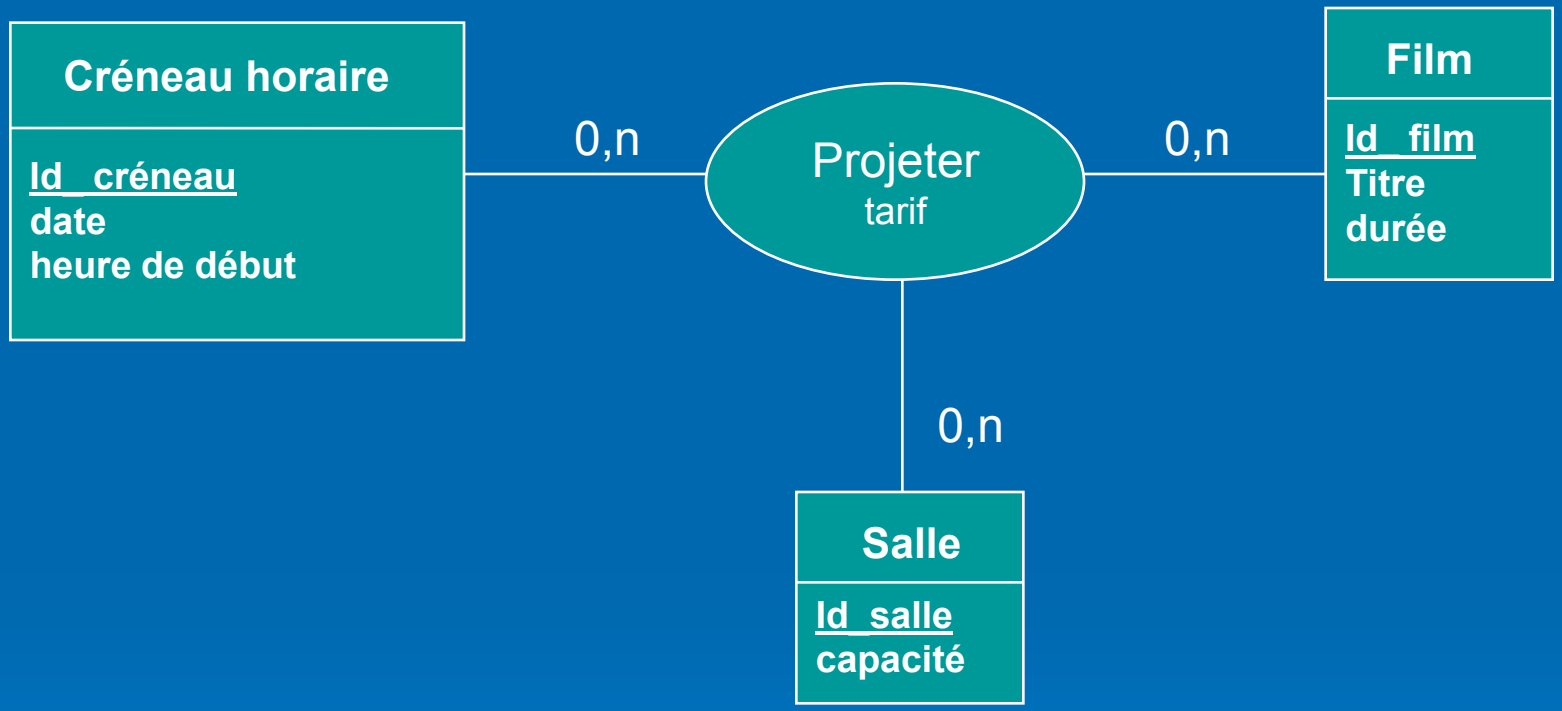
R8 : Association plusieurs ----- plusieurs
On crée toujours une table pour la relation en plus d'une table pour chacune des entités.



Immeuble(**Id_Immeuble**, etc.)
Propriétaire(**Id_Propriétaire**, etc.)

Immeuble(**Id_Immeuble**, etc.)
Propriétaire(**Id_Propriétaire**, etc.)
Appartient(**Id_Immeuble, Id_Propriétaire**)

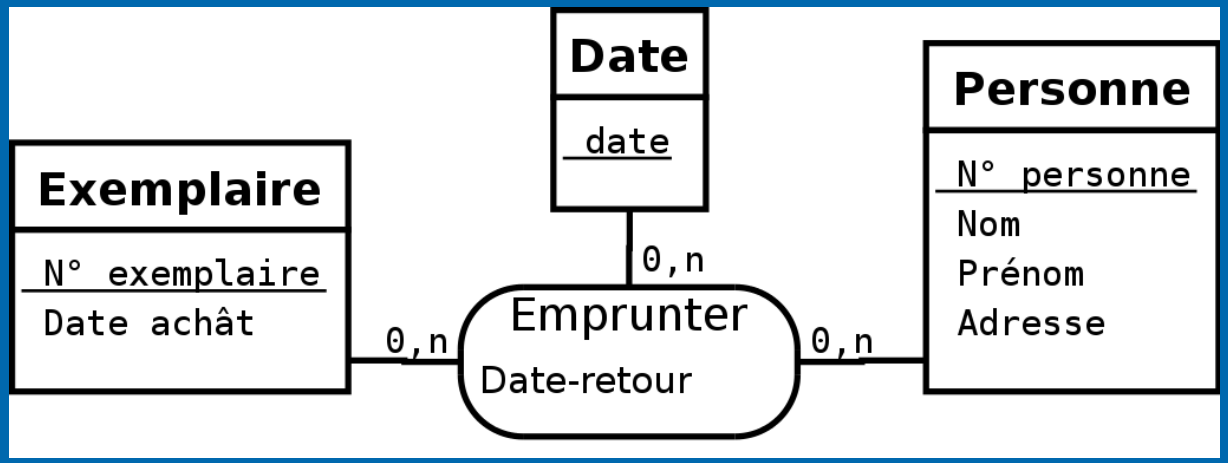
R9 : Association ternaire, premier cas : la relation entre chacune des paires d'entités ne peut déterminer la troisième entité. On crée une table pour chacune des entités et une table pour l'association. Cette dernière possèdera une clé primaire composée d'au moins trois champs.



Créneau(**Id_Créneau**, date,heure de début)
Film(**Id_Film**,titre,durée)
Salle(**Id_Salle**,capacité)
Projeter(**Id_Film, Id_Créneau, Id_Salle**,tarif)

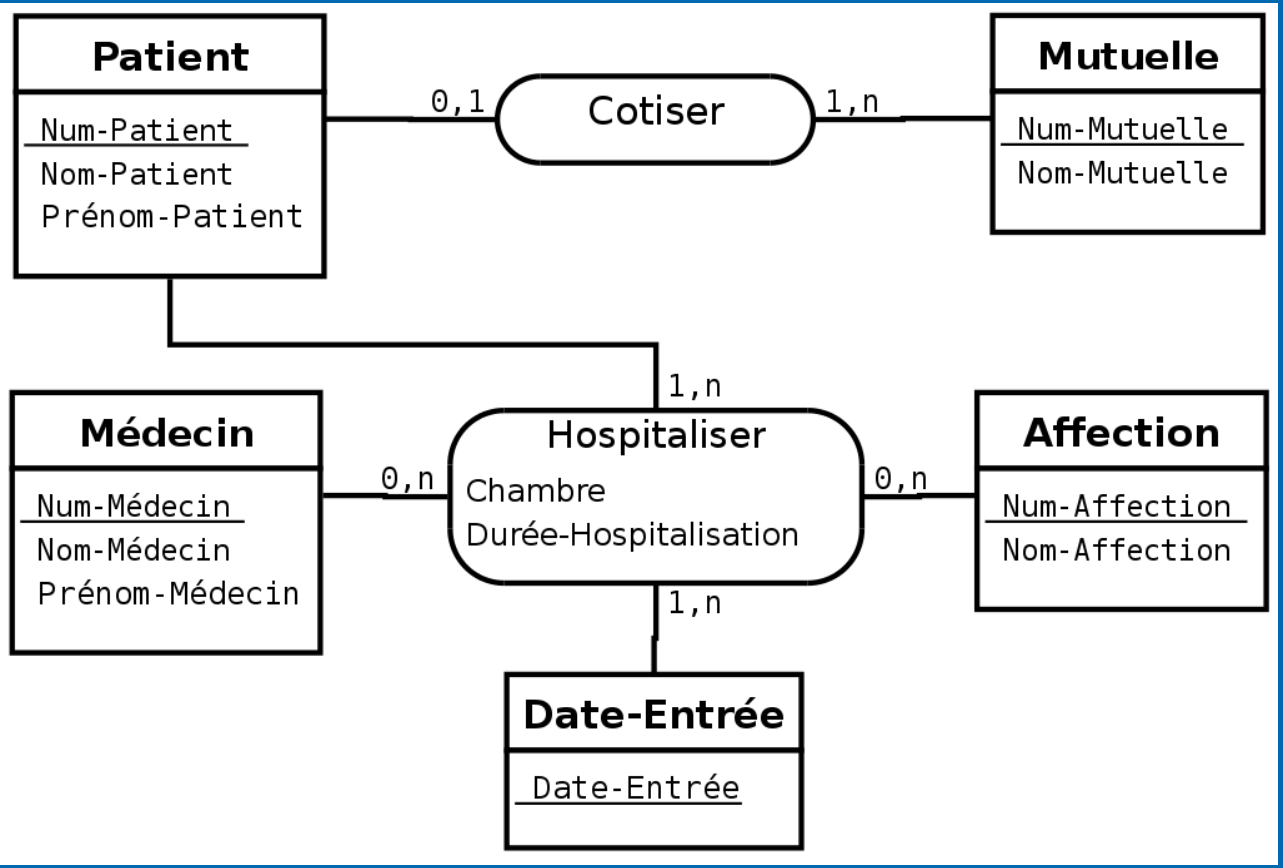
Cas particulier d'une entité sans attribut autre que sa clé

Lorsqu'une entité ne possède pas d'attribut en dehors de sa clé, il ne faut pas nécessairement en faire une relation.



- Exemplaire(**Num-Exemplaire**, date-achat)
- Personne(**Num-Personne**, nom, prénom, adresse)
- Emprunter(**Num-Exemplaire, Num-Personne, Date**, date-retour)

Exemple



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

Algèbre Relationnelle

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

On peut distinguer trois familles d'opérateurs relationnels

Les opérateurs unaires (Sélection, Projection)

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

Les opérateurs binaires ensemblistes (Union, Intersection Différence)

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

Les opérateurs binaires ou n-aires (Produit cartésien, Jointure, Division)

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

Sélection : *La sélection (parfois appelée restriction) génère une relation regroupant exclusivement toutes les occurrences de la relation R qui satisfont l'expression logique E, on la note* $\sigma_{(E)}R$.

Il s'agit d'une opération unaire essentielle dont la signature est :

relation × expression logique ⟶ relation

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

Exemple : Soit R(**Num**,Nom,Prénom)

$\sigma_{(Numéro>10)}R$

Num	Nom	Prénom
4	Lakhdar	Ammar
7	Mezhoud	Sami
13	Ferhad	Toufik
25	Baadeche	Mohamed

Num	Nom	Prénom
13	Ferhad	Toufik
25	Baadeche	Mohamed

Projection : *La projection consiste à supprimer les attributs autres que* ***A1, … An*** *d'une relation et à éliminer les* ***n-uplets en double*** *apparaissant dans la nouvelle relation ; on la note* ***Π(A1, … An)R.***

Il s'agit d'une opération unaire essentielle dont la signature est :

relation × liste d'attributs ⟶ relation

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

Exemple : Soit R(**Num**,Nom,Prénom)

Num	Nom	Prénom
4	Lakhdar	Ammar
7	Mezhoud	Sami
13	Ferhad	Toufik
25	Baadeche	Mohamed

$\Pi_{(Mat\ ,\ Nom)}R$

Num	Nom
4	Lakhdar
7	Mezhoud
13	Ferhad
25	Baadeche

$\Pi_{(Nom)}R$

Nom
Lakhdar
Mezhoud
Ferhad
Baadeche

Union : *L'union est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation constituée des n-uplets appartenant à chacune des deux relations R1 et R2* ***sans doublon****,*
on la note : ***R1 ∪ R2.***
Il s'agit une opération binaire ensembliste commutative essentielle dont la signature est :

relation × relation ⟶ relation

R1 et R2 ***doivent avoir les mêmes attributs*** *et si une même occurrence existe dans R1 et R2, elle n'apparaît qu'une seule fois dans le résultat de l'union. Le résultat de l'union est une nouvelle relation qui a les mêmes attributs que R1 et R2. Si R1 et R2 sont vides, la relation qui résulte de l'union est vide.*

R1(**Num**,Nom,Prénom)

Num	Nom	Prénom
4	Lakhdar	Ammar
13	Ferhad	Toufik
24	Messaadi	Ibtissem

R2(**Num**,Nom,Prénom)

Num	Nom	Prénom
4	Lakhdar	Ammar
20	Bouteldja	Mohamed
25	Baadeche	Mohamed

R1**U** R2 (**Num**,Nom,Prénom)

Num	Nom	Prénom
4	Lakhdar	Ammar
13	Ferhad	Toufik
20	Bouteldja	Mohamed
24	Messaadi	Ibtissem
25	Baadeche	Mohamed

Intersection : *L'intersection est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux appartenant aux deux relations,*
on la note ***R1 ∩ R2****.*
Il s'agit une opération binaire ensembliste commutative dont la signature est :

relation × relation ⟶ relation

*R*1 et *R*2 doivent avoir les mêmes attributs. Le résultat de l'intersection est une nouvelle relation qui a les mêmes attributs que *R*1 et *R*2. Si *R*1 ou *R*2 ou les deux sont vides, la relation qui résulte de l'intersection est vide.

R1(**Num**,Nom,Prénom) R2(**Num**,Nom,Prénom) R1 ∩ R2 (**Num**,Nom,Prénom)

Num	Nom	Prénom
20	Bouteldja	Mohamed
13	Ferhad	Toufik
7	Mezhoud	Sami

Num	Nom	Prénom
4	Lakhdar	Ammar
20	Bouteldja	Mohamed
25	Baadeche	Mohamed

Num	Nom	Prénom
20	Bouteldja	Mohamed

Différence : *La différence est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux ne se trouvant que dans la relation R1 ;*
on la note ***R1 − R2.***
Il s'agit une opération binaire ensembliste non commutative essentielle dont la signature est :

relation × relation ⟶ relation

*R*1 et *R*2 doivent avoir les mêmes attributs. Le résultat de la différence est une nouvelle relation qui a les mêmes attributs que *R*1 et *R*2. Si *R*1 est vide, la relation qui résulte de la différence est vide. Si *R*2 est vide, la relation qui résulte de la différence est identique à *R*1.

R1(**Num**,Nom,Prénom)

Num	Nom	Prénom
20	Bouteldja	Mohamed
13	Ferhad	Toufik
24	Messaadi	Ibtissem

R2(**Num**,Nom,Prénom)

Num	Nom	Prénom
4	Lakhdar	Ammar
20	Bouteldja	Mohamed
25	Baadeche	Mohamed

R1 - R2 (**Num**,Nom,Prénom)

Num	Nom	Prénom
13	Ferhad	Toufik
24	Messaadi	Ibtissem

Produit cartésien : *Le produit cartésien est une opération portant sur deux relations R1 et R2 et qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2, on la note* ***R1 × R2.***

Il s'agit une opération binaire commutative essentielle dont la signature est :

relation × relation ⟶ relation

Le résultat du produit cartésien est une nouvelle relation qui a tous les attributs de *R*1 et tous ceux de *R*2. Si *R*1 ou *R*2 ou les deux sont vides, la relation qui résulte du produit cartésien est vide. Le nombre d'occurrences de la relation qui résulte du produit cartésien est le nombre d'occurrences de *R*1 multiplié par le nombre d'occurrences de *R*2.

R1(**Num**,Nom,Prénom) R2(**Module**) R1 ***x*** R2 (**Num,Module**,Nom,Prénom)

Num	Nom	Prénom
20	Bouteldja	Mohamed
25	Baadeche	Mohamed

Module
SGBD
Math

Num	Nom	Prénom	Module
20	Bouteldja	Mohamed	SGBD
25	Baadeche	Mohamed	SGBD
20	Bouteldja	Mohamed	Math
25	Baadeche	Mohamed	Math

Jointure : *La jointure est une opération portant sur deux relations R1 et R2 qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2 qui satisfont l'expression logique E. La jointure est notée* $R1 \lhd\rhd_E R2$.

Il s'agit d'une opération binaire commutative dont la signature est :

relation × relation × expression logique ⟶ relation

Si *R*1 ou *R*2 ou les deux sont vides, la relation qui résulte de la jointure est vide.

En fait, la jointure n'est rien d'autre qu'un produit cartésien suivi d'une sélection :

$R1 \lhd\rhd_E R2 = \sigma_E (R1 \times R2)$

R1(**Num**,Nom,Prénom) R2(**Module**) $R1 \lhd\rhd_{(Module='SGBD')} R2$

Num	Nom	Prénom
20	Bouteldja	Mohamed
25	Baadeche	Mohamed

Module
SGBD
Math

Num	Nom	Prénom	Module
20	Bouteldja	Mohamed	SGBD
25	Baadeche	Mohamed	SGBD

Theta-jointure : *Une theta-jointure est une jointure dans laquelle l'expression logique E est une simple comparaison entre un attribut A1 de la relation R1 et un attribut A2 de la relation R2.*
La theta-jointure est notée $R1 \triangleleft\triangleright_E R2$.

Equi-jointure : *Une equi-jointure est une theta-jointure dans laquelle l'expression logique E est un test* ***d'égalité*** *entre un attribut A1 de la relation R1 et un attribut A2 de la relation R2. L'equi-jointure est notée* $R1 \triangleleft\triangleright_{A1,A2} R2$.

Jointure naturelle : *Une jointure naturelle est une equi-jointure dans laquelle les attributs des relations R1 et R2 portent le même nom A. Dans la relation construite,* ***l'attribut A n'est pas dupliqué mais fusionné en un seul attribut****.*
La jointure naturelle est notée $R1 \triangleleft\triangleright R2$.

Le résultat de la jointure naturelle est une nouvelle relation qui a tous les attributs de *R*1 et tous ceux de *R*2 sauf *A*. Il est en fait indifférent d'éliminer l'attribut *A* de la relation *R*1 ou *R*2.

Relation *Famille*

Nom	Prénom	Age
Fourt	Lisa	6
Juny	Carole	40
Fidus	Laure	20
Choupy	Emma	6

Relation *Cadeau*

Age	Article	Prix
40	livre	45
6	poupée	25
20	montre	87

Relation *R*

Nom	Prénom	Age	Article	Prix
Fourt	Lisa	6	poupée	25
Juny	Carole	40	livre	45
Fidus	Laure	20	montre	87
Choupy	Emma	6	poupée	25

Division : *La division est une opération portant sur deux relations R1 et R2, telles que le schéma de R2 est strictement inclus dans celui de R1, qui génère une troisième relation regroupant toutes les parties d'occurrences de la relation R1 qui sont associées à toutes les occurrences de la relation R2 ;*
on la note ***R1 ÷ R2.***
Il s'agit d'une opération binaire non commutative dont la signature est :

relation × relation ⟶ relation

Autrement dit, la division de *R*1 par *R*2 (*R*1 ÷ *R*2) génère une relation qui regroupe tous les n-uplets qui, concaténés à chacun des n-uplets de *R*2, donne toujours un n-uplet de *R*1.

La relation *R*2 ne peut pas être vide. Tous les attributs de *R*2 doivent être présents dans *R*1 et *R*1 doit posséder au moins un attribut de plus que *R*2 (inclusion stricte).

R1

Nom	Module
Bouteldja	SGBD
Baadeche	Math
Mezhoud	Math
Bouteldja	Math

R2

Module
SGBD
Math

*R*1 ÷ *R*2

Nom	Module
Bouteldja	SGBD
Bouteldja	Math

***R*1 ÷ *R*2 : Enseignant qui fait tous les modules**

Langage SQL

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

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

Langage de définition de données

Le langage de définition de données (LDD, ou *Data Definition Language*, soit DDL en anglais) est un langage orienté au niveau de la structure de la base de données. Le **LDD** permet de **créer, modifier, supprimer** des objets. Il permet également de **définir le domaine des données** (nombre, chaîne de caractères, date, booléen, ...) et d'ajouter des **contraintes** de valeur sur les données. Il permet enfin d'autoriser ou d'interdire l'accès aux données et d'activer ou de désactiver l'audit pour un utilisateur donné.
CREATE, ALTER, DROP, RENAME, ….

Langage de manipulation de données

Le langage de manipulation de données (LMD, ou *Data Manipulation Language*, soit DML en anglais) est l'ensemble des commandes concernant la manipulation des données dans une base de données. Le **LMD** permet **l'ajout, la suppression et la modification de lignes, la visualisation du contenu des tables et leur verrouillage**.

INSERT, UPDATE, DELETE, SELECT, ….

Langage de protections d'accès

Le langage de protections d'accès (ou *Data Control Language*, soit DCL en anglais) s'occupe de gérer les droits d'accès aux tables.

GRANT, REVOKE.

Langage de contrôle de transaction

Le langage de contrôle de transaction (ou *Transaction Control Language*, soit TCL en anglais) gère les modifications faites par le LMD, c'est-à-dire les caractéristiques des transactions et la validation et l'annulation des modifications.

COMMIT, SAVEPOINT, ROLLBACK, SET TRANSACTION

SQL intégré

Le SQL intégré (*Embedded SQL*) permet d'utiliser SQL dans un langage de troisième génération (C, Java, Cobol, etc.) : déclaration d'objets ou d'instructions ; exécution d'instructions ; gestion des variables et des curseurs ; traitement des erreurs.

LES REQUETES

L'Instruction SELECT

1- Cette requête peut être utilisée pour traduire une **PROJECTION (*Π(A1, … An)R*)**

SELECT champ1, champ2, ….
FROM table

SELECT A1,A2,….,An FROM R

Le caractère générique * veut dire tous les champs

Le nom de champ sert d'entête de colonne, pour utiliser un autre titre, appelé alias, on utilise l'instruction AS

SELECT DateN AS 'Date de naissance' FROM Etudiant

*Pour éliminer les **n-uplets en double on utilise l'instruction DISTINCT***
SELECT DISTINCT DateN FROM Etudiant

2 - Cette requête peut être utilisée pour traduire une **SELECTION (*σ(E)R.)***
Pour exprimer les condition, on utilise l'instruction WHERE

Exemple : Soit R(**Mat**,Nom,Prénom) $\sigma_{(Nom='Belhadj')}R$

```
SELECT *
 FROM R
 WHERE Nom='Belhadj'
```

Pour construire la condition E, on dispose des opérateurs : **= , > , < , >= , <= , <>**
Pour combiner plusieurs conditions on utilise les opérateurs logiques : **AND, OR, NOT**

Pour spécifier les conditions de recherche, on utilise les attributs suivants :

BETWEEN : utilisé pour vérifier si une valeur de colonne est entre deux valeurs

```
SELECT *
FROM Wilaya
WHERE Code_Wilaya BETWEEN 10 AND 25
```

IN : vérifie si une valeur de colonne correspond à l'une de celles spécifiées dans la liste.

NOT IN : vérifie si une valeur de colonne ne correspond pas à l'une de celles spécifiées dans la liste.

```
SELECT Nom, Ville
FROM Client
WHERE Ville NOT IN ('ALGER','ORAN')
```

LIKE : sélectionner les lignes en comparant la valeur d'une colonne de type Caractère à une chaîne de caractères spécifiée.

Le caractère % :

CC LIKE 'mot%' : CC est une chaîne de caractères qui commence par **'mot'**

CC LIKE '%mot%' : CC est une chaîne de caractères qui contient **'mot'**

```
SELECT Nom,Ville
FROM Client
WHERE Nom LIKE 'BEN%'
```

Utilisation d'expressions :

Dans la clause SELECT, on peut :
- Définir une nouvelle colonne (colonne calculée).
- Insérer des instruction descriptives qui seraient répétées à chaque ligne

```
SELECT Code,Désignation,Prix,TVA,Prix*TVA/100,'DA'
FROM Produit
```

```
SELECT Code,Désignation,Prix,TVA,Prix*TVA/100 AS 'Prix TTC','DA'
FROM Produit
```

Utilisation des fonction : Il existe trois principaux types de fonction en SQL :

Fonctions Mathématiques et Statistiques :

Count() : Compte le nombre d'enregistrements. Prend * comme argument.
Sum(Exp) : Calcule la somme des valeurs dans l'expression.
AVG(Exp) : Calcule la moyenne des valeurs dans l'expression.
Max(Exp) : Calcule la valeur maximale dans l'expression
Min(Exp) : Calcule la valeur minimale dans l'expression.

```
SELECT COUNT(*)
FROM Produit
```

```
SELECT MAX(Prix),MIN(Prix),AVG(Prix),SUM(Qte)
FROM Produit
```

Fonctions de dates :

Date	: Retourne la date courante.
Day(date)	: Retourne le quantième du mois (1 - 31).
Month(date)	: Retourne le mois (1 - 12).
Year(date)	: Retourne l'année (par exemple, 1994).

MapInfo

CurDate( )	: Retourne la date courante.
Weekday(date)	: Retourne le jour de la semaine (1 - 7). 1 représente le dimanche.

```
SELECT Code,YEAR(Date_Aq)
FROM Produit
```

Fonctions relatives aux caractères :

LEFT(chaîne,i) – RIGHT(chaîne,i) – LENGTH(chaîne) – SUBSTR(chaîne,i,j)

Classement des données :

La clause **ORDER BY** spécifie la liste des colonnes sur lesquelles l'affichage Doit être trié (**ASC ou DESC**)

```
SELECT *
FROM Produit
ORDER BY Désignation
```

Regroupement d'enregistrements :

La clause **GROUP BY** regroupe les lignes ayant des valeurs identiques dans une ou plusieurs colonnes issues de l'instruction SELECT en une ligne.

Le nombre de communes par wilaya:

```
SELECT CodeW,COUNT(*)
FROM Commune
GROUP BY CodeW
```

Clause HAVING:

La clause **HAVING** permet de restreindre la selection des groupes figurant dans le résultat. Elle précise la condition à laquelle chaque groupe doit répondre

Le nombre de communes par wilaya tel que le code wilaya > 10 :

```
SELECT CodeW,COUNT(*)
FROM Commune
GROUP BY CodeW
HAVING CodeW >10
```

Les Jointures :

La jointure entre deux ou plusieurs tables, peut être réalisée de deux manières.

La première :

- Citer les attributs recherchés dans la clause **SELECT**.
- Citer les différentes tables dans la clause **FROM**.
- Préciser la condition de Jointure dans la clause **WHERE**.

```
SELECT Nom,Prénom,Module
FROM Enseignant,Module
WHERE Module='SIG'
```

La deuxième :

Utiliser la clause **JOIN … ON …**

```
SELECT Nom,Prénom,Module
FROM Enseignant JOIN Module ON Module='SIG'
```

Les requêtes imbriquées :

On utiliser des requêtes imbriquées pour joindre deux requêtes.
Ex: Les communes de la wilaya d'ALGER. (sans avoir le code de la wilaya)

```
SELECT CodeCommune,NomCom,NbrHabitant
FROM Commune
WHERE CodeW = (SELECT CodeW
               FROM Wilaya
               WHERE NomWilaya='ALGER')
```

```
SELECT Nom,Prénom,Age
FROM Joueur
WHERE CodeJoueur IN (SELECT CodeJoueur
                     FROM JouerEquipe
                     WHERE Equipe='JSK')
```

Requêtes d'insertion :

Pour insérer des données dans une table, on utilise la clause **INSERT INTO**
INSERT INTO <nom table> [(nom col1, nom col2,…)] VALUES (val1,val2,…)

INSERT INTO Module (CodeM,Module)
VALUES ('SIG','Système d'Information Géographique')

Requêtes de modification :

Pour modifier des données dans une table, on utilise la clause **UPDATE**
UPDATE <nom table> SET nom col1=Val1, nom col2=val2… [WHERE <cond>]

UPDATE Commune SET Classe='3' WHERE NombrHab>40000

Requêtes de suppression :

Pour supprimer des lignes d'une table, on utilise la clause **DELETE**
DELETE FROM <nom table> WHERE <nom conl> = <Valeur>

DELETE FROM Commune WHERE NombrHab=0