COURS
CONCEPTION DE BASE DE DONNEES

Mr MANE

Licence 2 Télécom 2015-2016
ESTM

Présentation

- ✓ Etroitement lié au cours Base de Données Relationnelles
- ✓ La conception des bases de données est la tâche la plus ardue du processus de développement du système d’information.
- ✓ Recourir à une méthode de conception afin de faciliter la communication et la coopération entre les différents acteurs d’une organisation

Auteurs et Métiers

Exprimer des besoins relatifs à un domaine d'application

Utilisateur

Analyste

Administrateur

Production et interrogation De la BD

Développeur

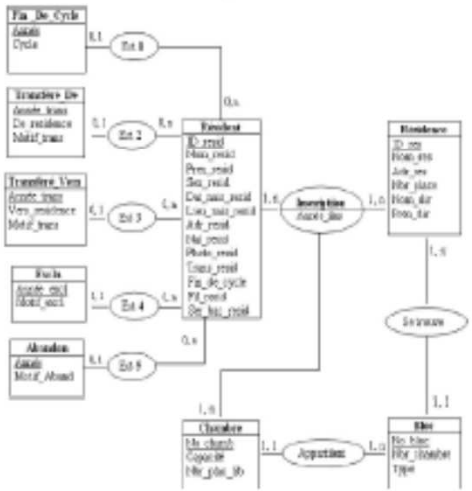


Schéma conceptuel

Objectif du cours

Objectifs :

- Analyser les besoins.
- Apprendre à organiser les données en fonction des besoins.
- Appliquer une méthodologie de conception de BD
- Appréhender la théorie des bases de données

Plan du cours

Introduction

1. Modélisation conceptuelle de donné (MCD)

2. Modèle Logique de données (MLD)

3. Modèle physique de données (MPD)

Introduction

Un système de gestion de bases de données **SGBD** : Le système logiciel qui permet à des utilisateurs de définir, créer, mettre à jour une base de données et d'en contrôler l'accès.
Un **SGBD** est formé par:

- ✓ La base de données: Collection de données cohérentes ayant une signification précise
- ✓ Le catalogue: Contient la définition ou la description de la **BD** (structure de chaque fichier, format de stockage et type de chaque élément de données, divers contraintes sur les données...). Les informations stockées dans le catalogue sont appelés métadonnées. Les métadonnées sont les données sur les données. Tout ce qui décrit la base de données, par opposition au contenu de la base de données, sont des métadonnées. Ainsi, les noms de colonnes, les noms de bases de données, les noms d'utilisateur, les noms de version sont des métadonnées.
- ✓ Le **SGBD** : Constitue par l'ensemble des programmes qui vont permettre la description, l'implémentation physique ainsi que la manipulation des données de la **BD**.

Les **SGBD** peut être résumée en distinguant quatre générations de modèles de base de données:

Première génération (années 60)

- ✓ Séparation de la description des données et des programmes d'application
- ✓ Traitement de fichiers reliés par des structures de graphe
- ✓ SGBD IDMS

Deuxième génération (années 70)

- ✓ Simplification du SGBD externe
- ✓ Modèle relationnel
- ✓ Langage non procédural
- ✓ SGBD ORACLE, INFORMIX, …

Troisième génération (années 80)

SGBD objet

- ✓ SGBD Relationnel objet : ORACLE 8
- ✓ SGBD orienté objet : O2

Quatrième génération (années 90)

- ✓ Bases de données et internet
- ✓ Entrepôts de données (data warehouse)
- ✓ Fouille de données (data mining)

Quand nous construisons directement les tables d'une base de données dans un système de gestion dés bases de données (**SGBD**)(Oracle, SQL Server, DB2, Access, MySQL, PostGre, ...), nous sommes exposés à deux types de problème :

- ✓ nous ne savons pas toujours dans quelle table placer certaines colonnes (par exemple, l'adresse de livraison se met dans la table des clients ou dans la table des commandes?) ;
- ✓ nous avons du mal à prévoir les tables de jonction intermédiaires (par exemple, la table des interprétations qui est indispensable entre les tables des films et la table des acteurs).

Il est donc nécessaire de recourir à une étape préliminaire de conception. Les techniques présentées ici font partie de la méthodologie Merise (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise) élaboré en France en 1978, qui permet notamment de concevoir un système d'information d'une façon standardisée et méthodique.

Le but de ce support de cours est d'introduire le schéma entités-associations (section 1), le schéma relationnel (sections 2 et 3) et d'expliquer la traduction entre les deux (sections 2.3 et 4). La construction du schéma entités-associations peut se faire en étudiant les dépendances fonctionnelles (section 1.3)) et en tenant compte les contraintes, les traitements, le langage relationnel

MODELE CONCEPTUEL DE DONNEES (MCD)

1. Définition

Le modèle conceptuel des données (MCD) a pour but de représenter de façon structurées les données qui seront utilisées par le système d'information. Le modèle conceptuel des données décrit la sémantique c'est-à-dire le sens attaché à ces données et à leurs rapport et non à l'utilisation qui peut en être faite.

1.1 Schéma entités-associations

La modélisation conceptuelle que nous proposons dans ce document pour un univers dont on veut stocker les données, conduit à l'élaboration d'un type de schéma très répandu, le Schéma entités-associations

1.1.1 Entités et associations

Une **entité** est une population d'individus homogènes. Par exemple, les produit ou les articles vendus par une entreprise peuvent être regroupés dans une même entité **articles** (figure 1) , car d'un article à l'autre, les informations ne changent pas de nature (à chaque fois, il s'agit de la désignation, le prix unitaire, etc.)

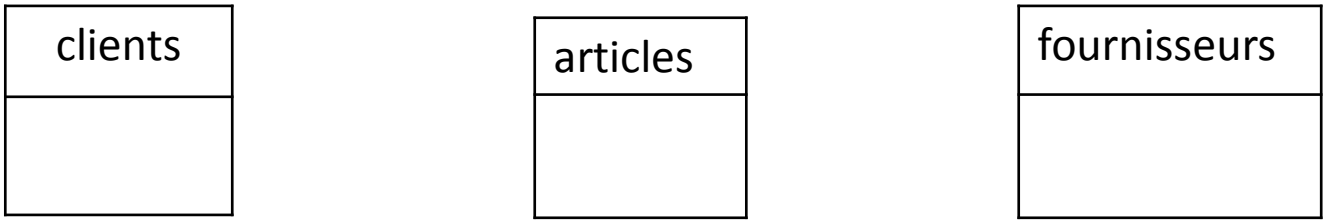


FIG. 1 – Entités

Par contre, les articles et les clients ne peuvent pas être regroupés: leurs information ne sont pas homogènes (un article ne possède pas d'adresse et un client ne possède pas de prix unitaire). Il faut donc leur réserver deux entités distinctes: l'entité ***articles*** et l'entité ***clients***

Une **association** un lien entre deux entités (ou plus). On doit lui donner un nom, souvent un verbe; qui caractérise le type de relation entre les **entités.** Dans notre exemple, l'association commander est une liaison évidente entre les entités articles et clients.

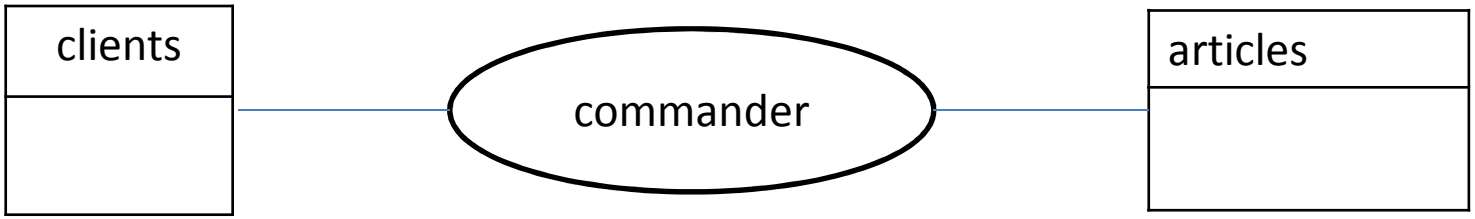


FIG. 2 – Association

1.1.2 Attributs et identifiants

Un attribut est une propriété d'une entité ou d'une association
Dans notre exemple, le prix unitaire est un attribut de l'entité **article**s, nom est un attribut de l'entité ***clients*** , la quantité commandée est un attribut de l'association ***commander***

FIG. 3 – Attributs

Ensuite, chaque individu d'une entité doit être identifiable de manière unique. C'est pourquoi toutes les entités doivent posséder un attribut sans doublon (c'est-à-dire ne prenant pas deux fois la même valeur). Il s'agit de l'identifiant que l'on souligne sur le schéma, par convention. Le numéro de client constitue un **identifian**t classique pour l'entité clients

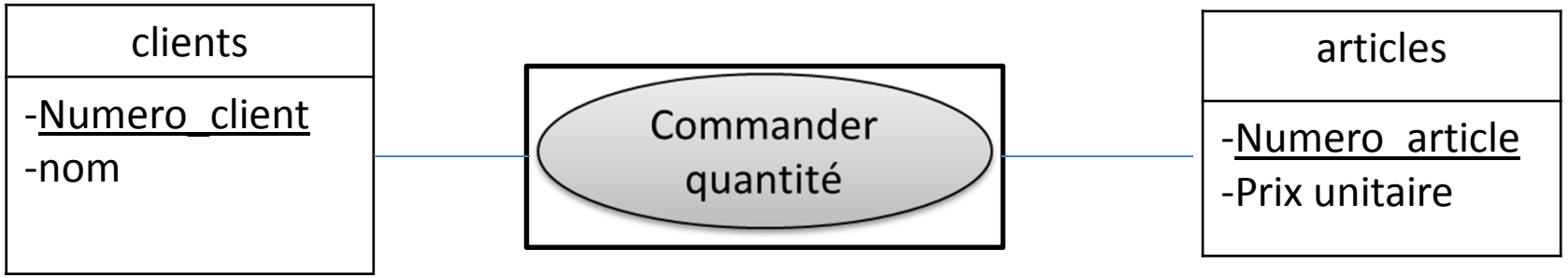


FIG. 4 – Identifiant

Remarques:

- ✓ Une entité possède au moins un attribut (son identifiant);
- ✓ Au contraire, une association peut être dépourvue d'attribut.

1.1.3 Cardinalités

Ce sont des expressions qui permettent d'indiquer combien de fois au minimum et au maximum le lien entre 2 entités peut se produire. Pour une association de 2 entités, il y a 4 cardinalités à indiquer.
Il y a trois valeurs typiques: 0, 1 et N (plusieurs).
Les cardinalités traduisent des règles de gestion. Ce sont des règles propre au SI étudié qui exprime des contraintes sur le modèle.
Exemple:

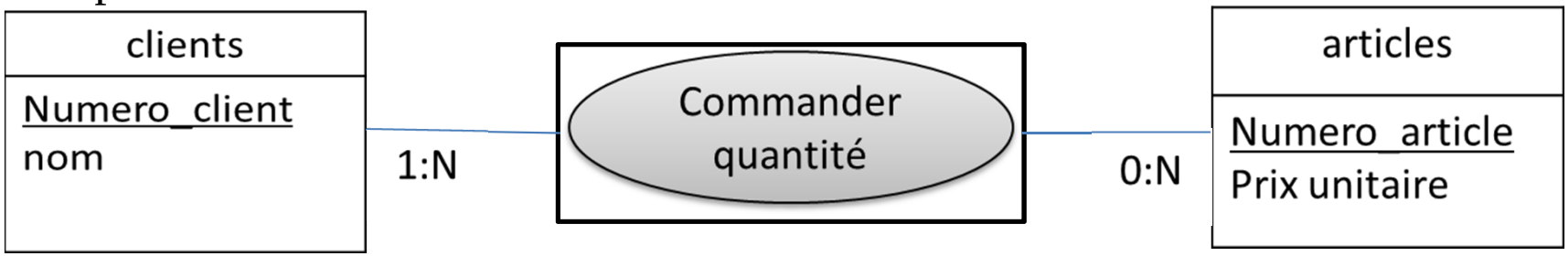


FIG. 5 – Cardinalités

La seule difficulté pour établir correctement les cardinalités est de poser les questions dans le bon sens. Autour de l'association commander , par exemple:

- ✓ côté clients, la questions est «un client peut commander combien d'article? » et la réponse est «entre 1 et plusieurs »;
- ✓ côté articles, la question est «un article peut être commandé par combien de client? » et cette fois-ci, la réponse est «entre 0 et plusieurs ».

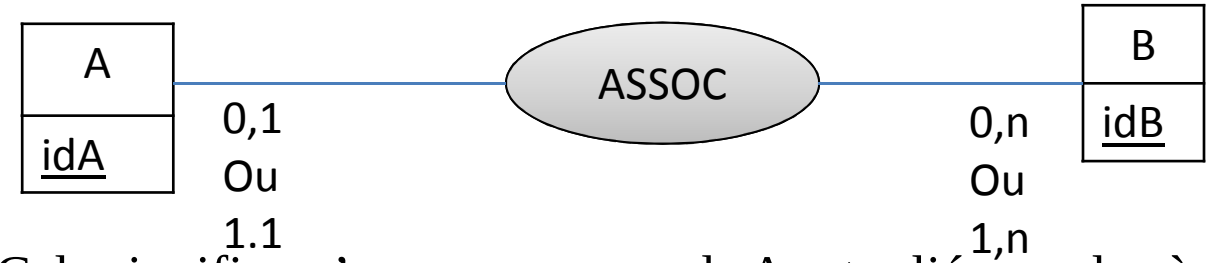
1.1.4 Les associations binaires concernant 2 entités

On distingue trois catégories d'associations en fonction des cardinalités maximales des ses branches:

- Les associations hiérarchiques encore appelées [1,n] ou associations fonctionnelles
- Les associations non hiérarchiques encore appelées [n,n] ou associations non fonctionnelles
- Les associations [1,1] , les 2 branches ont pour cardinalité maximale 1, Ce cas est rare,

a. Les associations hiérarchiques [1, n]

Ce sont les associations ou d'un côté la cardinalité maximale est à 1 de l'autre côté la cardinalité maximale est à n.



- Cela signifie qu'une occurrence de A est reliée au plus à une occurrence de B. C'est-à-dire si on connaît une occurrence de A alors on aura forcément quelle est la seule occurrence de B qui correspond (si elle existe). On dit que A détermine B. C'est un lien de dépendance fonctionnelle. B dépend fonctionnellement de A

L’entité qui correspond à la branche du coté du 1 est parfois appelée entités fils et l’entité correspondant à la branche du côté n est parfois appelée entités père. Cette appellation découle de l’analogie: un fils n’a qu’un seul père, et père peut avoir plusieurs fils.

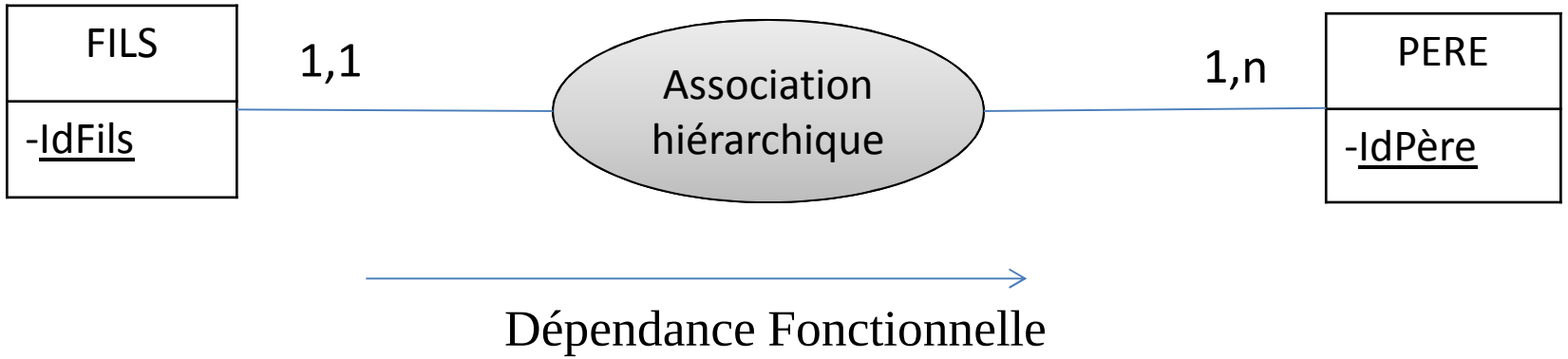


FIG. 6 – associations hiérarchiques

b. Les associations non hiérarchiques [n, n]

c. Les autres types d’association

Mais une association peut aussi relie 3 ou très rarement 4 entités (quasiment jamais plus). On parle d’association ternaire, quaternaire (n-aire).

Enfin, une association peut aussi relier une entités à elle-même, c'est-à-dire que des occurrences de la même entité sont reliées entre elles. L'association correspondante est qualifiée de réflexive.
Ce sont ces autres types d'association que nous allons étudier maintenant

1. Les associations réflexives

Une association réflexive est une association reliant des occurrences de la même entité. Ces associations sont quasiment toujours binaire (2 branche).
Pour lire une association réflexive, il est faut connaitre **le rôle à chaque branche de l'association.**
Il existe comme pour les autres associations réflexives hiérarchique et des associations réflexives non hiérarchique

1.1 Exemple associations réflexives hiérarchiques

Dans une entreprise bureaucrate, chaque salarié a un seul chef (supérieur hiérarchique direct) sauf le patron et chaque chef a au moins au salarié sous ses ordres. Evidement, certains salariés ne sont chefs de personnes

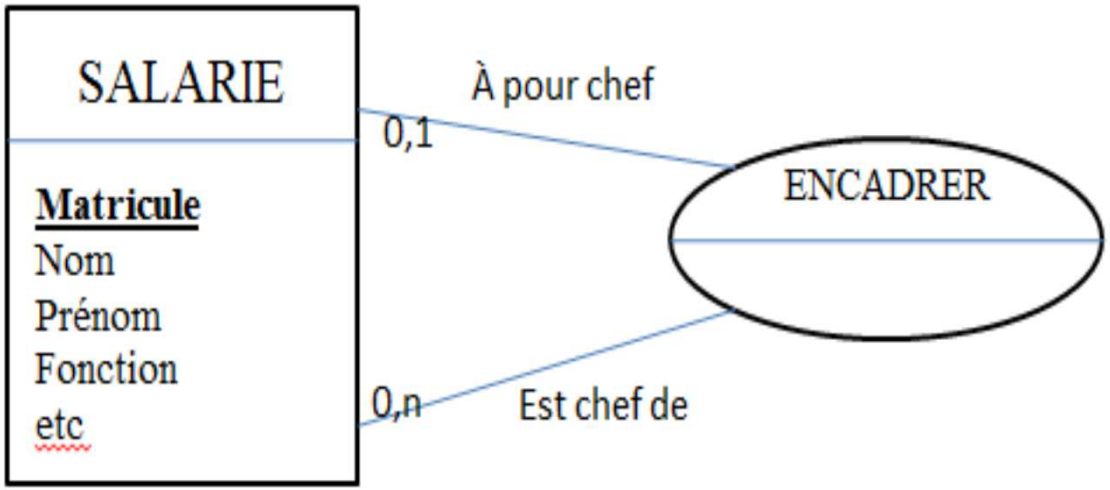


FIG. 7 – associations réflexives hiérarchiques

1.2 Associations réflexives non hiérarchiques

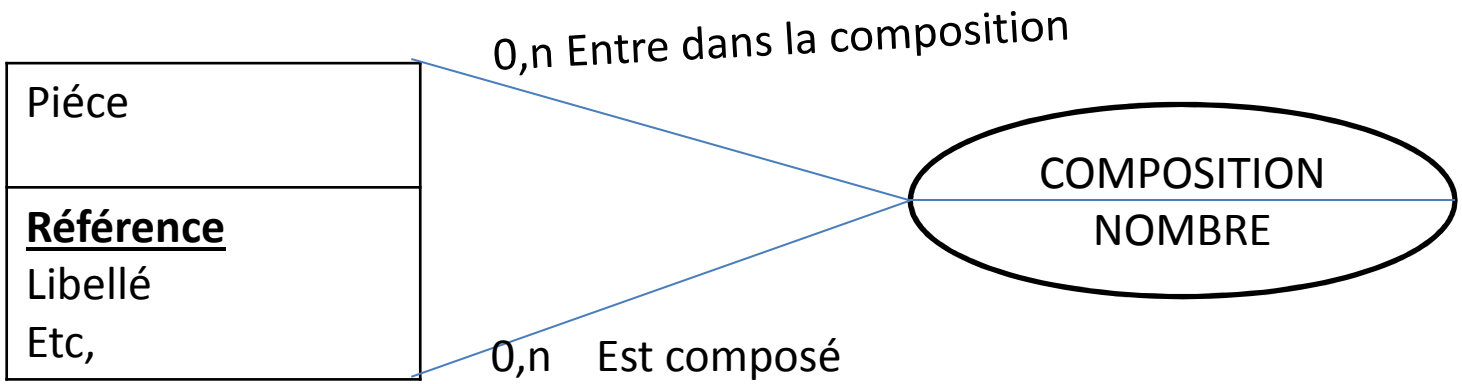


FIG. 8 – associations réflexives non hiérarchiques

Pièces composants, composé=réflexive asymétrique

2. Les associations de dimension 3 ou plus (ternaires ou plus)

Une association peut relier plus de 2 entités ensemble, le plus souvent trois. On parle alors d'association ternaire. On utilise une association ternaire quand on a besoin de connaitre une occurrence de chaque entité pour avoir une information.

Exemple:

Pour connaître la quantité de chacune des pièces fabriquée par chanque ouvrier à une date données, on utilise une association ternaire entre OUVRIER,PIECE et date. La quantité est une données portée par cette association

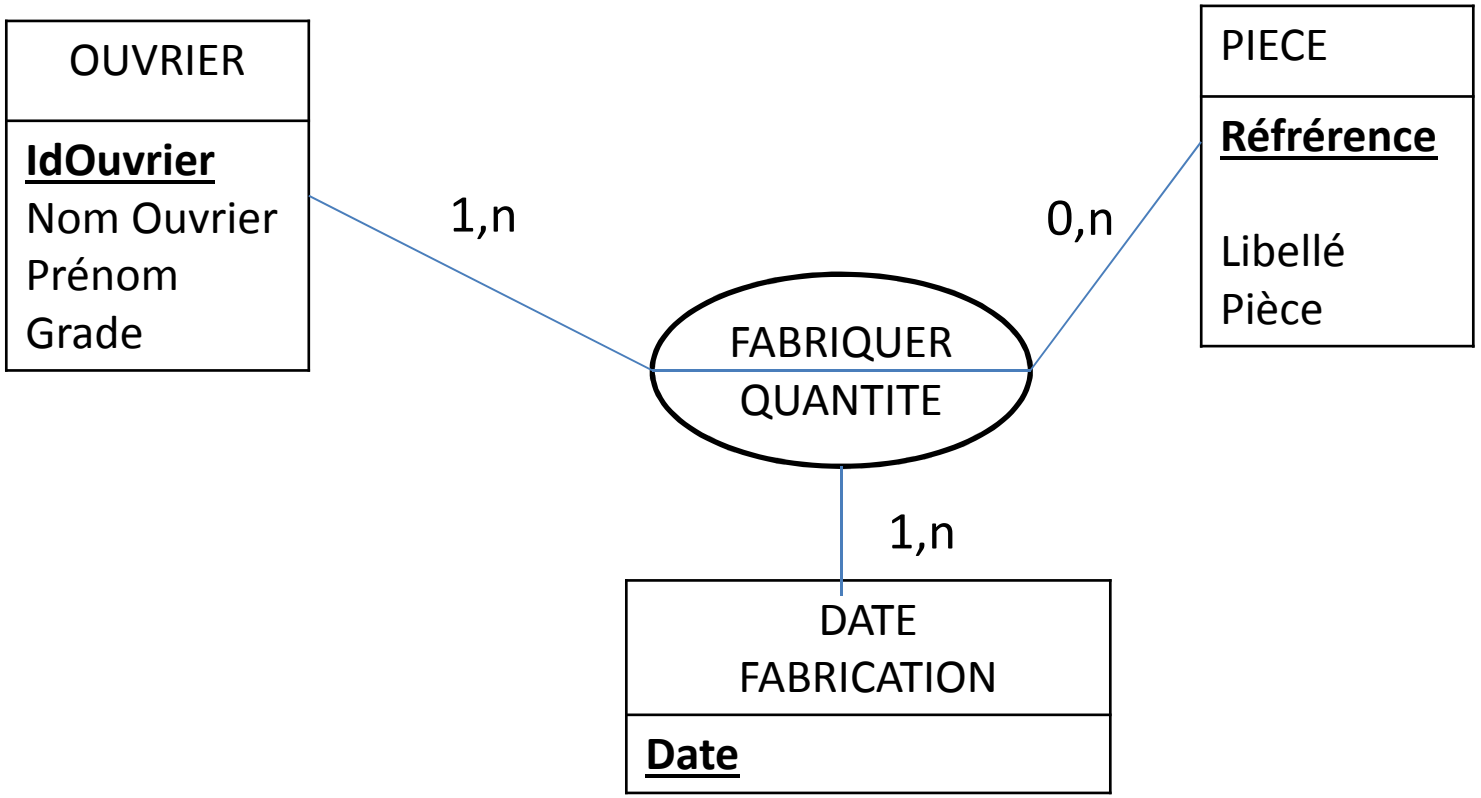


FIG. 9 – associations ternaire

Une occurrence de l'association Fabrique implique une seule occurrence de chaque entité

L’association ternaire implique aussi que:

Pour un ouvrier, on peut avoir plusieurs pièces différentes à la même date (il peut fabriquer plusieurs types de pièces le même jour).
Une pièce peut être fabriquée par plusieurs ouvriers différents le même jour.
A des dates différentes, un même ouvrier peut fabriquer les mêmes pièces.

L’identifiant d’une association ternaire est formée de la concaténation (juxtaposition) les identifiants des entités reliées.

1.2 Dépendances fonctionnelles

Pour établir efficacement un modèle entités-associations bien normalisé, on peut étudier au préalable les dépendances fonctionnelles entre les attributs puis, les organiser en graphe de couverture minimale. Cette technique est traditionnellement employé pour normaliser des schémas relationnels, mais elle s'applique très bien en amont, au niveau des modèles conceptuels.

1.2.1 Définitions et propriétés

Un attribut Y dépend fonctionnellement d'un attribut X si et seulement si une valeur de X induit une unique valeur de Y . On note une dépendance fonctionnelle par une flèche simple : $X \rightarrow Y$.

Par exemple, si X est le numéro de client et Y le nom de client, alors on a bien $X \rightarrow Y$. Par contre, on a pas $Y \rightarrow X$, car plusieurs clients de numéros différents peuvent porter le même nom.

Transitivité : si X → Y et Y → Z alors X → Z.

Par exemple, on a numéro de commande → numéro de client → nom de client, donc on a aussi numéro de commande → nom de client. Mais la dépendance fonctionnelle numéro de commande → nom de client est dite *transitive*, car il faut passer par le numéro de client pour l'obtenir.
Au contraire, la dépendance fonctionnelle numéro de client → nom de client est directe .

Il y a Deux types de dépendance fonctionnelle:

Dépendance fonctionnelle élémentaire

Une **dépendance fonctionnelle** (DF) X—›Y est élémentaire,
s'il n'existe pas Z incluse dans X qui assure elle-même une
DF: Z—›Y.

Exemple:

1.Ref_Article—›Nom_Article VRAI
2.Num_Facture,Ref_Article—›Qte_Article VRAI
3.Num_Facture,Ref_Article—›Nom_Article FAUX
car Ref_Article—›Nom_Article

Dépendance fonctionnelle directe

Une dépendance fonctionnelle (DF) X—›Y est directe, s'il n'existe pas un attribut Z (ou une collection d’attributs) qui engendrerait une DF transitive X—›Z—›Y.

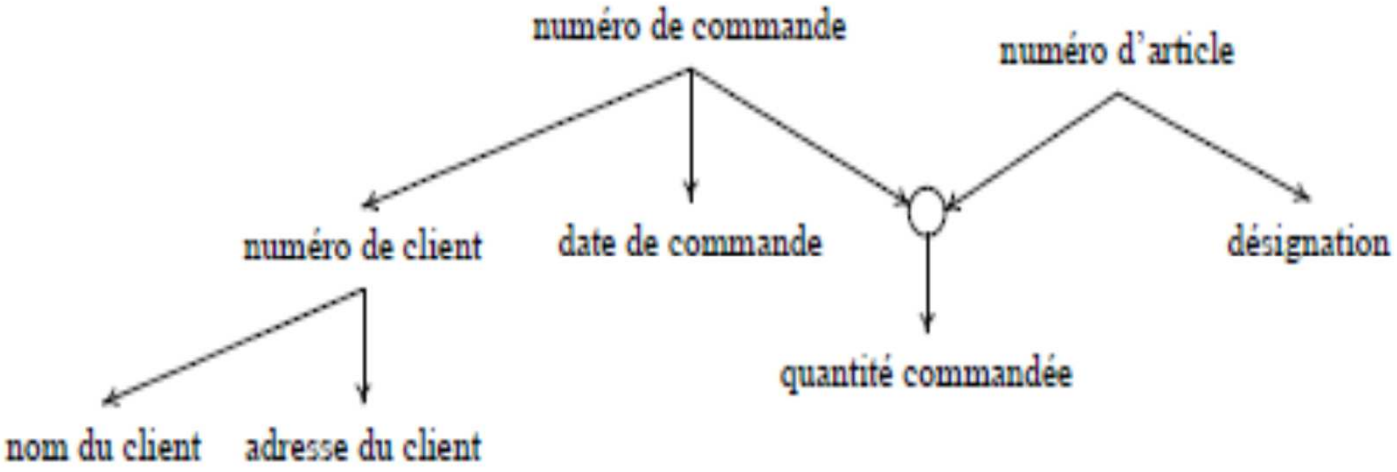
Exemples:

1.Num_Facture—›Num_Représentant
2.Num_Représentant—›Nom_Représentant
3.Num_Facture—›Nom_Représentant

DF non directe car on peut écrire:
Num_Facture—›Num_Représentant—›Nom_Représentant

1.2.2 Graphe de couverture minimale

En représentant tous les attributs et toutes les dépendances fonctionnelles directes entre eux, nous obtenons un réseau appelé graphe de couverture minimale. Dans notre exemple sur les clients, les commandes et les articles, ce graphe est donné sur la figure.



La technique de traduction en un schéma entités-associations qui suit, suppose qu'aucun attribut n'a été oublié sur le graphe de couverture minimal et notamment, aucun identifiant. D'ailleurs toutes les dépendances fonctionnelles du graphe doivent partir d'un identifiant. Si ce n'est pas le cas, c'est qu'un identifiant a été omis.

1.3 Règles de normalisation

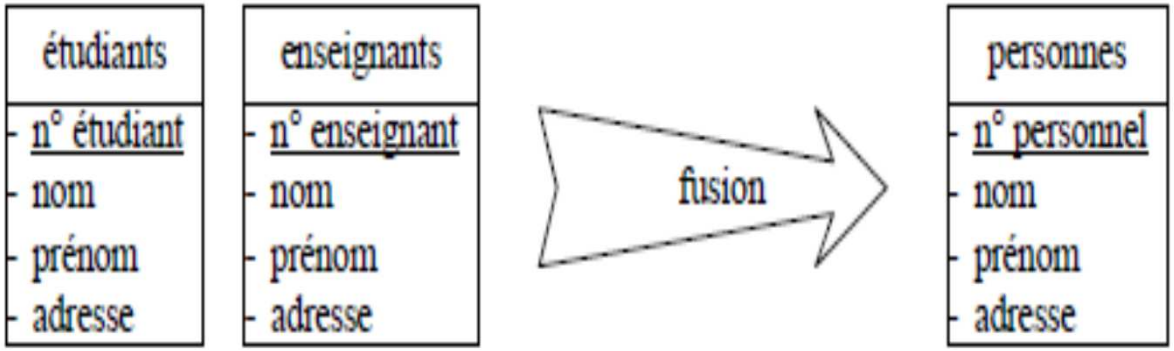
Un bon schéma entités-associations doit répondre à 9 règles de normalisation, que le concepteur doit connaître par cœur.

1.3.1 Les bonnes manières dans un schéma entités-associations

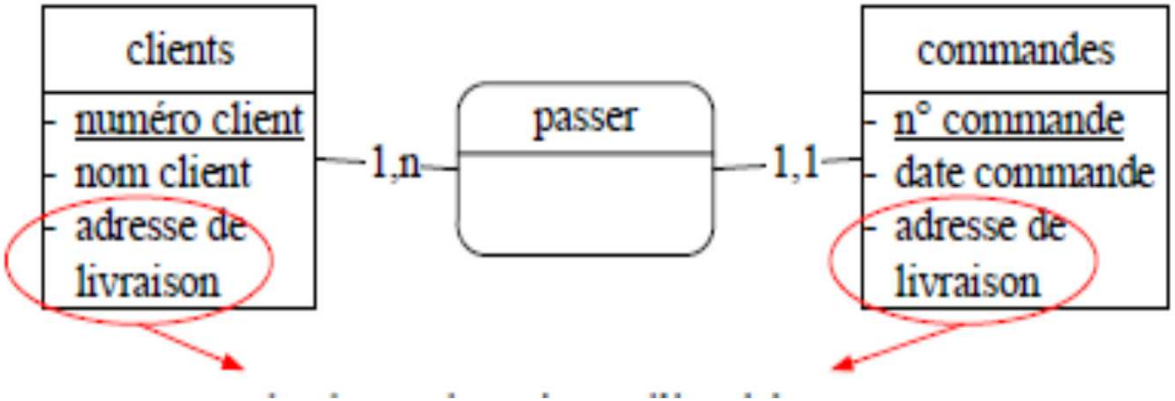
Normalisation des entités (importante): toutes les entités qui sont remplaçables par une association doivent être remplacées.
Normalisation des noms: le nom d'une entité, d'une association ou d'un attribut doit être unique.
Conseils:

- ✓ Pour les entités, utiliser un nom commun au pluriel (par exemple: clients);
- ✓ Pour les associations, utiliser un verbe à l'infinitif (par exemple: effectuer, concerner) éventuellement à la forme passive (être commandé) et accompagné d'un adverbe (avoir lieu dans, pendant, à):
- ✓ Pour les attributs, utiliser un nom commun singulier (par exemple: nom, libellé, description) éventuellement accompagné du nom de l'entité ou de l'association dans laquelle il se trouve (par exemple: nom de client, numéro d'article).

Remarque: Lorsqu'il reste plusieurs fois le même nom, c'est parfois symptomatique d'une modélisation qui n'est pas terminée (figure 10(a)) ou le signe d'une redondance (figure 11(b))



(a) Deux entités homogènes peuvent être fusionnées



Redondance, donc risque d'incohérence

(b) Si deux attributs contiennent les mêmes informations, alors la redondance induit non seulement un gaspillage d'espace mais également un grand risque d'incohérence: ici, les adresses risquent de ne pas être les mêmes et dans ces conditions, ou faut-il livrer?

Normalisation des identifiants : chaque entité doit posséder un identifiant.

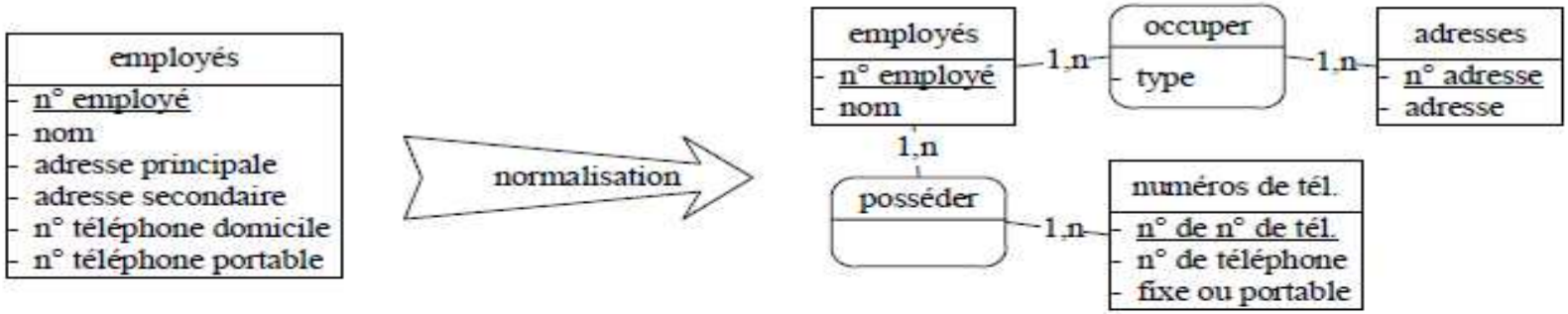
Conseils :

- ✓ éviter les identifiants composés de plusieurs attributs (comme par exemple un identifiant formé par les nom et prénom), car d'une part c'est mauvais pour les performances et d'autres part, l'unicité supposée par une telle démarche finit tôt ou tard par être démentie ;
- ✓ préférer un identifiant court pour rendre la recherche la plus rapide possible (éviter notamment les chaînes de caractères comme un numéro de plaque d'immatriculation, un numéro de sécurité sociale ou un code postal)
- ✓ éviter également les identifiants susceptibles de changer au cours du temps (comme les plaques d'immatriculation ou les numéros de sécurité sociale provisoires).

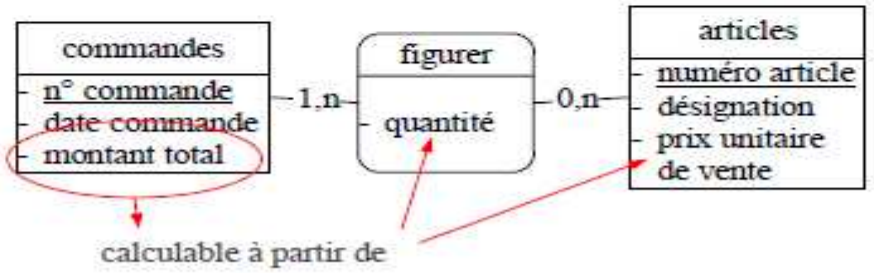
Conclusion: l'identifiant sur un schéma entités-associations (et donc la future clé primaire dans le schéma relationnel) doit être un entier, de préférence incrémenté automatique.

Normalisation des attributs (importante) : remplacer les attributs en plusieurs exemplaires en une association supplémentaire de cardinalités maximales **n** et ne pas ajouter d'attribut calculable à partir d'autres attributs.

En effet, d'une part, les attributs en plusieurs exemplaires posent des problèmes d'évolutivité du modèle (sur la figure 11(a) à gauche, comment faire si un employé a deux adresses secondaires ?) et



(a) Attributs en plusieurs exemplaires remplacés par une association supplémentaire



(b) Attribut calculable qu'il faut retirer du schéma

FIG . 11– Contre-exemples de la normalisation des attributs

d'autre part, les attributs calculables induisent un risque d'incohérence entre les valeurs des attributs de base et celles des attributs calculés, comme sur la figure11(b).

D’autres d’attributs calculables classiques sont à éviter, comme l’âge (qui est calculable à partir de la date de naissance) ou encore le département (calculable partir d’une sous-chaîne du code postal).

Normalisation des attributs des associations (importante) : les attributs d’une association doivent dépendre directement des identifiants de toutes les entités en association.

Par exemple, la quantité commandé dépend à la fois du numéro de client et du numéro d’article, par contre la date de commande non. Il faut donc faire une entité commandes à part, idem pour les livraisons.

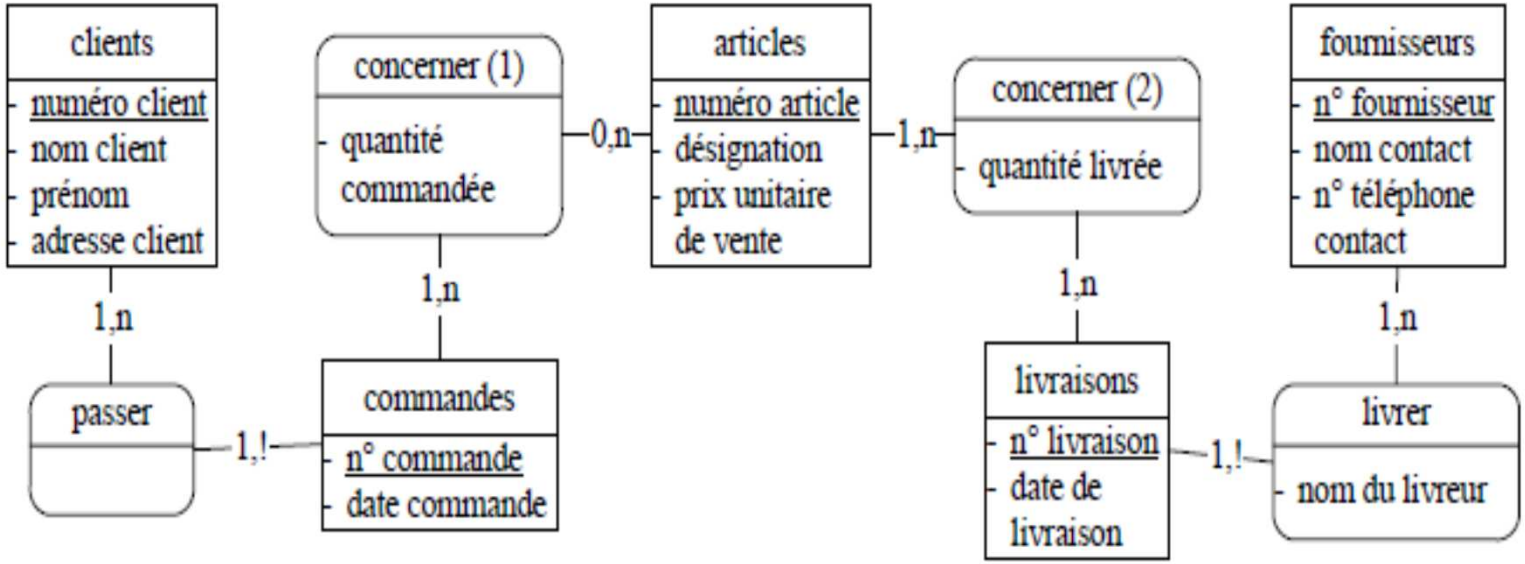


Fig. 12 – Normalisation des attributs des associations

L'inconvénient de cette règle de normalisation est qu'elle est difficile à appliquer pour les associations qui ne possèdent pas d'attribut. Pour verifier malgré tout qu'une association sans attribut est bien normalisé, on peut donner temporairement cette association un attribut imaginaire (mais pertinent) qui permet de vérifier la règle.

Par exemple, entre les entités livres et auteurs de la figure15, l'association écrire ne possède pas d'attribut. Imaginons que nous ajoutions un attribut pourcentage qui contient le pourcentage du livre écrit par chaque auteur (du même livre). Comme cet attribut pourcentage dépend à la fois du numéro de livre et du numéro d'auteur, l'association écrire est bien normalisé.

Autre conséquence de la normalisation des attributs des associations : une entité avec une cardinalité de 1,1 ou 0,1 aspire les attributs de l'association.

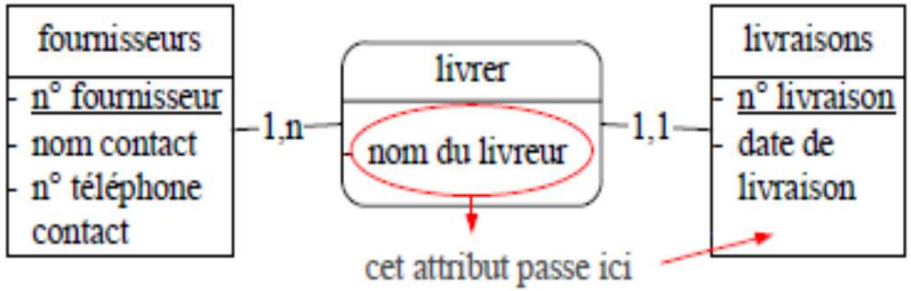
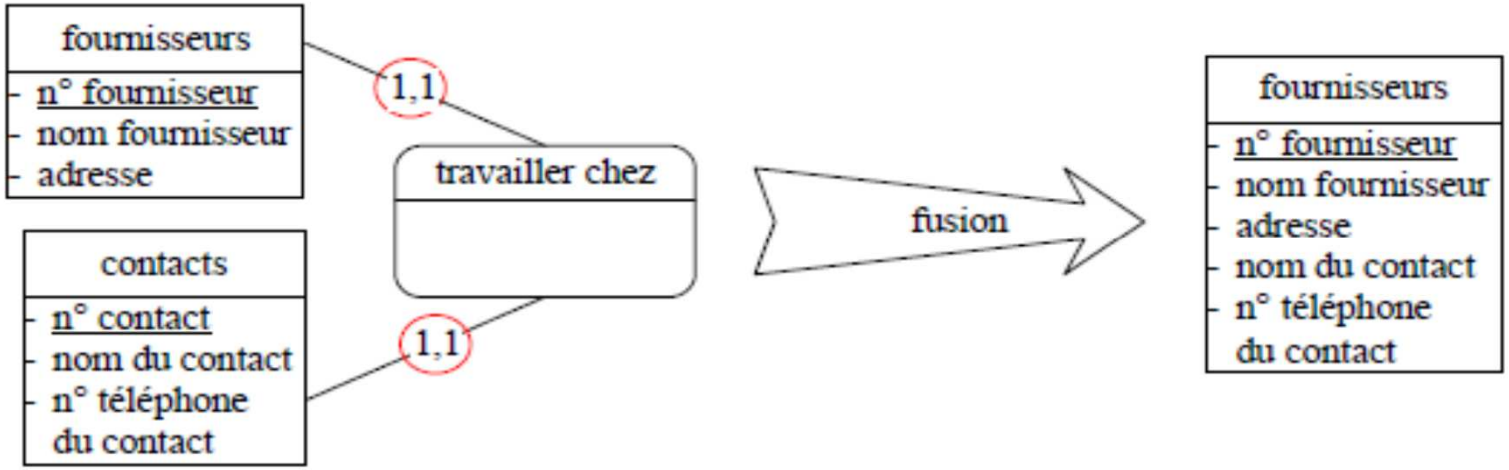


Fig. 13 – Cardinalité 1,1 et attributs d'une association

Normalisation des associations (importante) : il faut éliminer les associations fantômes (figure 14(a)),redondantes (figure 14(b)) ou en plusieurs exemplaires (figure 14(c)).



(a) les cardinalités sont toutes 1,1 donc c'est une association fantôme

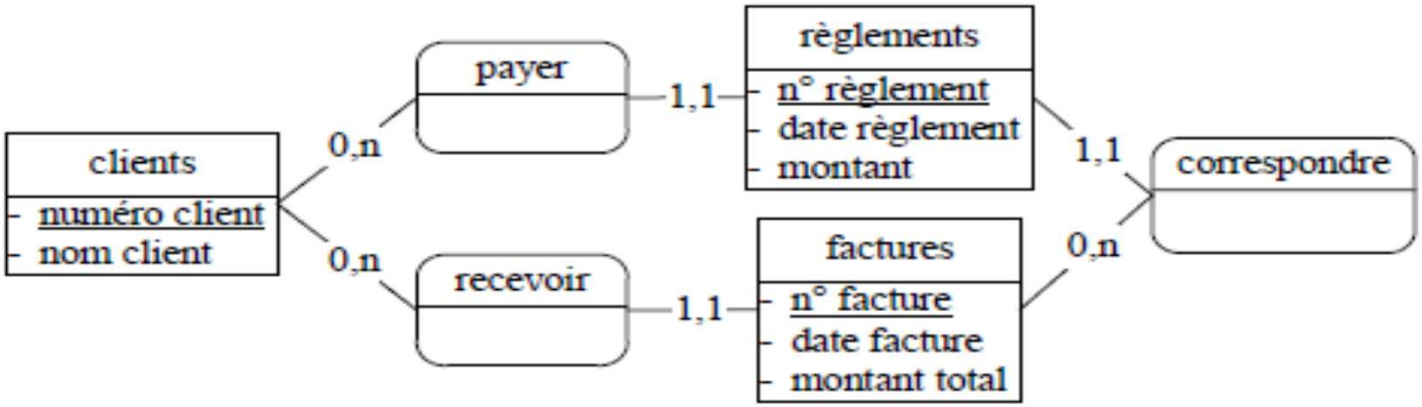


Fig. 14 – Contre-exemples de la normalisation des associations

(b) si un client ne peut pas régler la facture d'un autre client, alors l'association payer est inutile et doit être supprimé (dans le cas contraire, l'association payer doit être maintenue)

En ce qui concerne les associations redondantes, cela signifie que s'il existe deux chemins pour se rendre d'une entité une autre, alors ils doivent avoir deux significations ou deux durées de vie différente. Sinon, il faut supprimer le chemin le plus court, car il est déductible partir de l'autre chemin. Dans notre exemple de la figure 14(b), si on supprime l'association payer, on peut retrouver le client qui a payé le règlement en passant par la facture qui correspond.

Remarque : une autre solution pour le problème de la figure 14(b) consiste retirer l'entité règlement et d'ajouter une association régler avec les mêmes attributs (sauf l'identifiant) entre les entités clients et factures.

Normalisation des cardinalités : une cardinalité minimale est toujours 0 ou 1 (et pas 2, 3 ou n) et une cardinalité maximale est toujours 1 ou n (et pas 2, 3, ...).

Cela signifie que si une cardinalité maximale est connue et vaut 2, 3 ou plus, alors nous considérons quand même qu'elle est indéterminé et vaut n. Cela se justifie par le fait que même si nous connaissons n au moment de la conception, il se peut que cette valeur évolue au cours du temps. Il vaut donc mieux considérer n comme une inconnue dés le départ.

1.3.2 Les formes normales

La normalisation est un processus qui consiste à convertir les structures de données complexes en structures de données simples et stables.

Le processus de normalisation est réalisé en étapes. Chaque étape correspond à une application d'une forme normal.

Il existe environ huit (8) formes normales mais seulement les trois (3) premières sont les plus utilisées.

Première forme normale

Une table est en 1FN seulement, et seulement si:

elle possède une clé primaire qui sert à identifier chaque ligne de façon unique; et,

la valeur de chaque attribut est atomique (indivisible, non décomposable)

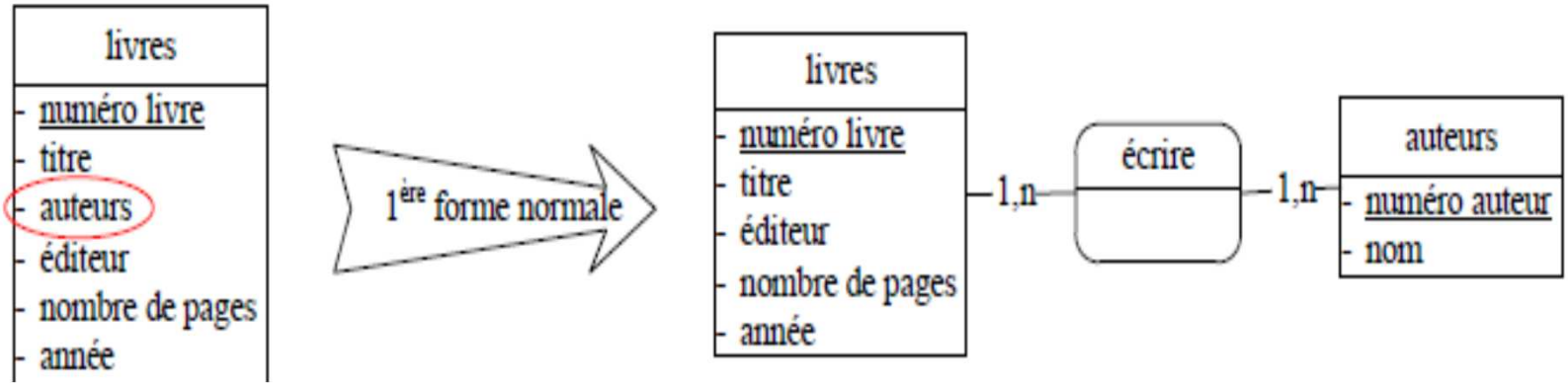
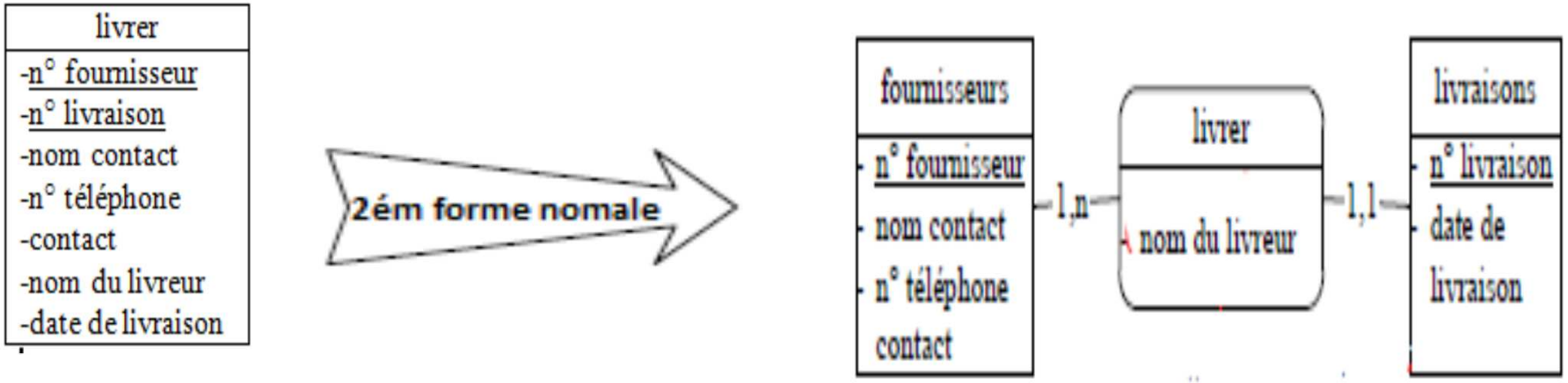


Fig. 15 – Application de la première forme normale : il peut y avoir plusieurs auteurs pour un livre donnée

Deuxième forme normale

Une table est en 2FN seulement, et seulement si: elle est en 1FN; et tous ses attributs non-clés dépendent complètement de la clé primaire (i.e., pas de dépendance partielle).



Troisième forme normale de Boyce-Codd

Une table est en 3Fn seulement, et seulement si:

- ✓ elle est en 2FN; et,
- ✓ on n'y trouve aucune dépendance transitive entre les attributs non clés et la clé primaire
- ✓ Tout attribut n'appartenant pas à une clé ne dépend pas d'un autre attribut non clé

Si ce n'est pas le cas, il faut placer l'attribut pathologique dans une entité séparé, mais en association avec la première.

numéro avion	constructeur	modèle	capacité	propriétaire
1	Airbus	A380	180	Air France
2	Boeing	B747	314	British Airways
3	Airbus	A380	180	KLM

Tab. 1 – Il y a redondance (et donc risque d'incoherence) dans les colonnes constructeur et capacité

Par exemple, l'entité avions (figure 16 ci dessous gauche) dont les valeurs sont données dans le tableau ci-dessus n'est pas en troisième forme normale de Boyce-Codd, car la capacité et le constructeur d'un avion ne d´pendent pas du numéro d'avion mais de son modèle. La solution normalisée est donnée figure 16 droite.

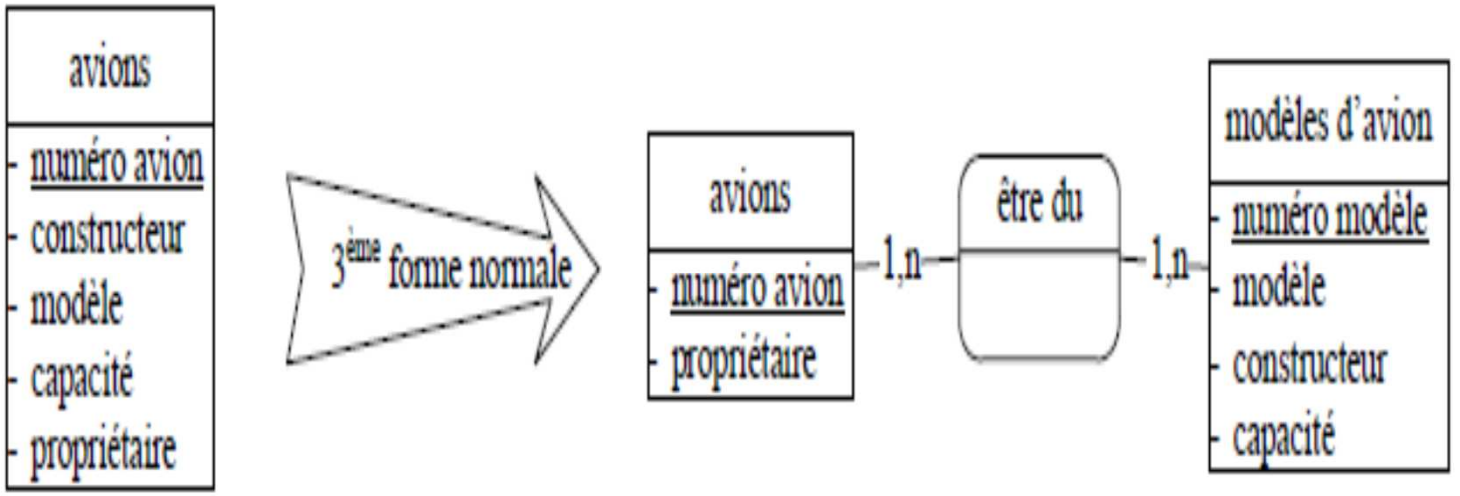


Fig. 15 – Application de la troisième forme normale de Boyce-Codde

1.4 Méthodologie de base

Face à une situation bien définie (soit à travers un énoncé précis, soit à travers une collection de formulaire ou d'états que le nouveau système d'information est censé remplacer), nous pouvons procéder sans établir le graphe de couverture minimale:

- ✓ identifier les entités en présence ;
- ✓ lister leurs attributs ;
- ✓ ajouter les identifiants (numéro arbitraire et auto-incrémenté) ;

- ✓ établir les associations binaires entre les entités ;
- ✓ lister leurs attributs ;

- ✓ calculer les cardinalités ;

- ✓ vérifier les règles de normalisation et en particulier, la normalisation des entités (c'est à ce stade qu'apparaissent les associations non binaires), des associations et de leurs attributs ainsi que la troisième forme normale de Boyce-Codd ;
- ✓ effectuer les corrections nécessaires.

Mais, il est parfois plus intuitif d'en passer par l'étude des dépendances fonctionnelles directes :

- ✓ identifier les entités en présence et leur donner un identifiant (numéro arbitraire et auto-incrémenté) ;
- ✓ ajouter l'ensemble des attributs et leur dépendances fonctionnelles directes avec les identifiants (en commençant par les dépendances élémentaires) ;

- ✓ À ce stade, la majorité des règles de normalisation devraient être vérifiés, il reste tout de même la normalisation des noms, la présence d'attributs en plusieurs exemplaires et d'associations redondantes ou en plusieurs exemplaires, à corriger.

Il faut garder également à l'esprit que le modèle doit être exhaustif (c'est-à-dire contenir toutes les informations nécessaires) et éviter toute redondance qui, on ne le dira jamais assez, constitue une perte d'espace, une démultiplication du travail de maintenance et un risque d'incohérence.

Il faut par ailleurs veiller à éliminer les synonymes (plusieurs signifiants pour un signifie, exemple : nom, patronyme, appellation) et les polysèmes (plusieurs signifiés pour un signifiant, exemples : qualité, statut).

2. Modèle logique de données (MLD)

Maintenant que le **MCD** est établi, on peut le traduire en différents systèmes logiques et notamment les bases de données relationnelles qui proposent une vision plus concrète pour modéliser la situation.

2.1 Modèle logique relationnel

2.1.1 Tables, lignes et colonnes

Lorsque des données ont la même structure (comme par exemple, les renseignements relatifs aux clients), on peut les organiser en table dans laquelle les colonnes décrivent les champs en commun et les lignes contiennent les valeurs de ces champs pour chaque enregistrement (tableau 2).

Numéro client	nom	prénom	adresse
1	Diop	Paul	10, rue Grd Dakar
2	Sall	Jean	22, rue Médina
3	Diallo	Mamadou	Fass
4	Ka	Ousmane	Hlm 3
…	…	…	…

TAB.2 - Contenu de la table clients, avec en première ligne les intitulés des colonnes

2.1.2 Clés primaires et clés étrangères

Les lignes d'une table doivent être uniques, cela signifie qu'une colonne (au moins) doit servir à les identifier. Il s'agit de la clé primaire de la table.
L'absence de valeur dans une clé primaire ne doit pas être autorisée. Autrement dit, la valeur vide (NULL) est interdite dans une colonne qui sert de clé primaire, ce qui n'est pas forcément le cas des autres colonnes, dont certaines peuvent ne pas être renseignés à toutes les lignes.

De plus, la valeur de la clé primaire d'une ligne ne devrait pas, en principe, changer au cours du temps.

Par ailleurs, il se peut qu'une colonne Colonne1 d'une table ne doive contenir que des valeurs prises par la colonne Colonne2 d'une autre table (par exemple, le numéro du client sur une commande doit correspondre à un vrai numéro de client). La Colonne2 doit être sans doublons (bien souvent il s'agit d'une clé primaire). On dit alors que la Colonne1 est clé étrangère et qu'elle référence la Colonne2

Par convention, on souligne les clés primaires et on fait précéder les clés étrangères d'un dièse # dans la description des colonnes d'une table :

clients(numéro client, nom client, prénom, adresse client
commandes(numéro commande, date de commande, #numéro client (non vide))

Remarques :

- ✓ une même table peut avoir plusieurs clés étrangères mais une seule clé primaire (éventuellement composées de plusieurs colonnes) ;
- ✓ une colonne clé étrangère peut aussi être primaire (dans la même table) ;
- ✓ une clé étrangère peut être composée (c'est le cas si la clé primaire référencée est composée) ;
- ✓ implicitement, chaque colonne qui compose une clé primaire ne peut pas recevoir la valeur vide (NULL interdit) ;
- ✓ par contre, si une colonne clé étrangère ne doit pas recevoir la valeur vide, alors il faut le préciser dans la description des colonnes.

Les SGBDR vérifient au coup par coup que chaque clé étrangère ne prend pas de valeurs en dehors de celles déjà prises par la ou les colonne(s) qu'elle référence. Ce mécanisme qui agit lors de l'insertion, de la suppression ou de la mise à jour de lignes dans les tables, garantit ce que l'on appelle l'intégrité référentielle des données.

2.1.3 Schéma relationnelle

On peut représenter les tables d'une base de données relationnelle par un schéma relationnel dans lequel les tables sont appelées relations et les liens entre les clés étrangères et leur clé primaire est symbolisé par un connecteur

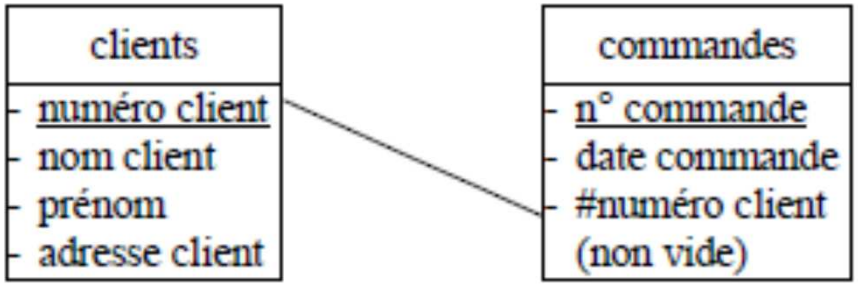


Schéma relationnel simple entre deux tables

Certains éditeur inscrivent sur le connecteur un symbole 1 côté clé primaire et un symbole ∞ côté clé étrangère (à condition que celle-ci ne soit pas déjà clé primaire). Il faut prendre garde avec cette convention, car le symbole ∞ se trouve du côté opposé à la cardinalité maximale n correspondante.

2.2 Traduction d'un MCD en un MLDR

Pour traduire un MCD en un MLDR, il suffit d'appliquer cinq règles.

Notations : on dit qu'une association binaire (entre deux entités ou réflexive) est de type :

- ✓ 1 : 1 (un à un) si aucune des deux cardinalités maximales n'est n ;
- ✓ 1 : n (un à plusieurs) si une des deux cardinalités maximales est n ;
- ✓ n : m (plusieurs à plusieurs) si les deux cardinalités maximales sont n.

En fait, un schéma relationnel ne peut faire la différence entre 0,n et 1,n. Par contre, il peut la faire entre 0,1 et 1,1 (règles 2 et 4).

Règle 1 : toute entité devient une table dans laquelle les attributs deviennent les colonnes. L'identifiant de l'entité constitue alors la clé primaire de la table.
Par exemple, l'entité articles figure 12 devient la table :

articles(<u>numéro article</u>, désignation, prix unitaire de vente)

Règle 2 : une association binaire de type 1 : n disparaît, au profit d'une clé étrangère dans la table, côté 0,1 ou 1,1 qui référence la clé primaire de l'autre table. Cette clé étrangère ne peut pas recevoir la valeur vide si la cardinalité est 1,1.

Par exemple, l'association livrer figure 12 est traduite par :

fournisseurs(n° fournisseur, nom contact, n° téléphone contact)
livraisons(n° livraison, date de livraison, nom livreur, #n° fournisseur (non vide))

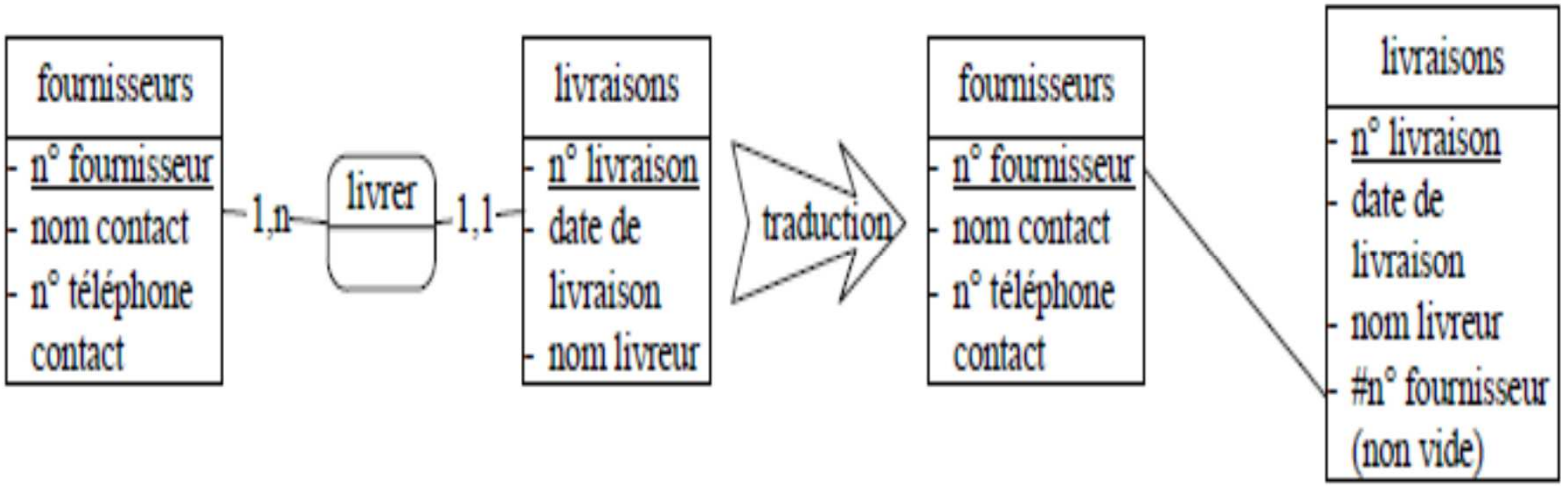
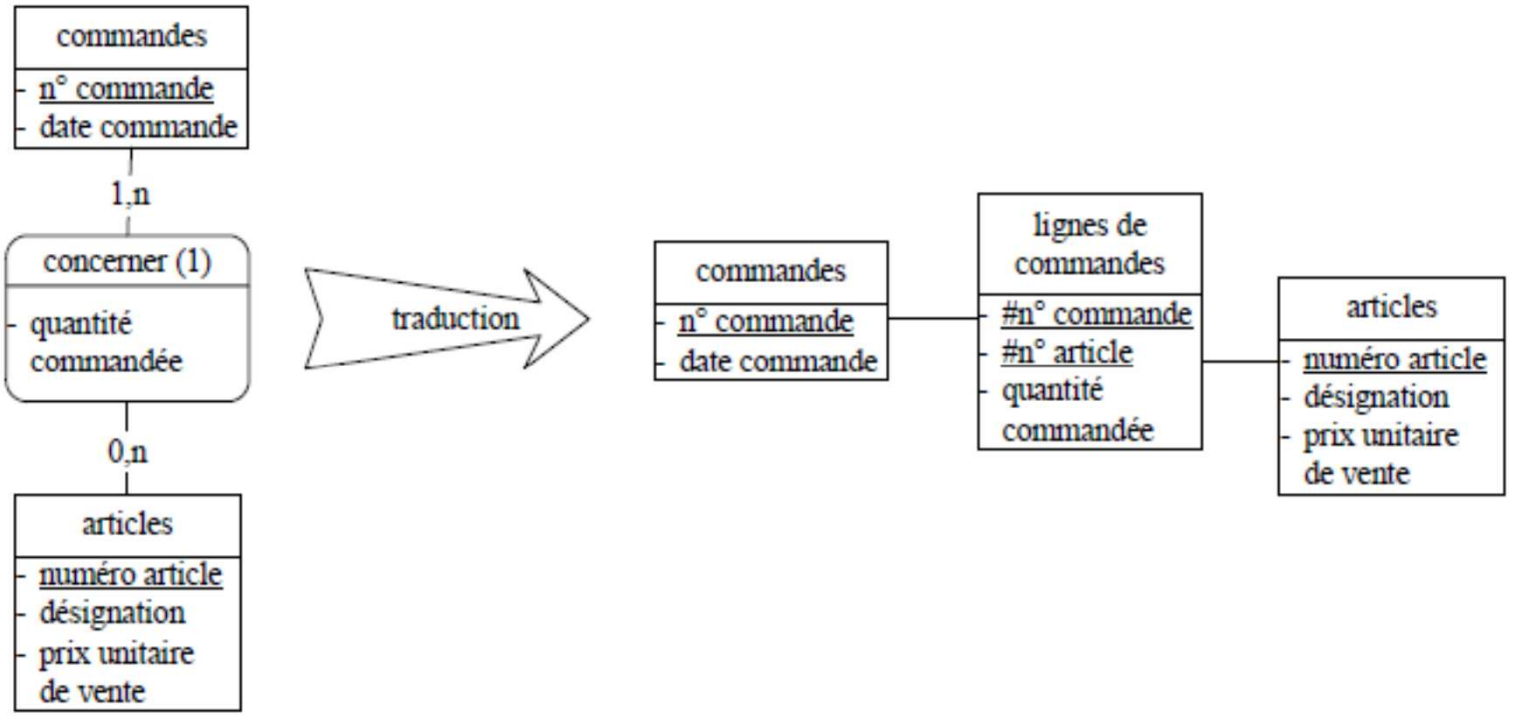


FIG -Traduction d'une association de type 1 : n

Il ne devrait pas y avoir d'attribut dans une association de type 1 : n, mais s'il en reste, alors ils glissent vers la table côté 1.

Règle 3 : une association binaire de type n : m devient une table supplémentaire (parfois appelé table de jonction, table de jointure ou table d'association) dont la clé primaire est composée de deux clés étrangères (qui référencent les deux clés primaires des deux tables en association). Les attributs de l'association deviennent des colonnes de cette nouvelle table.

Par exemple, l'association concerner (1) de la figure 12 est traduite par la table supplémentaire lignes de commande :
lignes de commande(#n° commande, #n° article, quantité commandée)

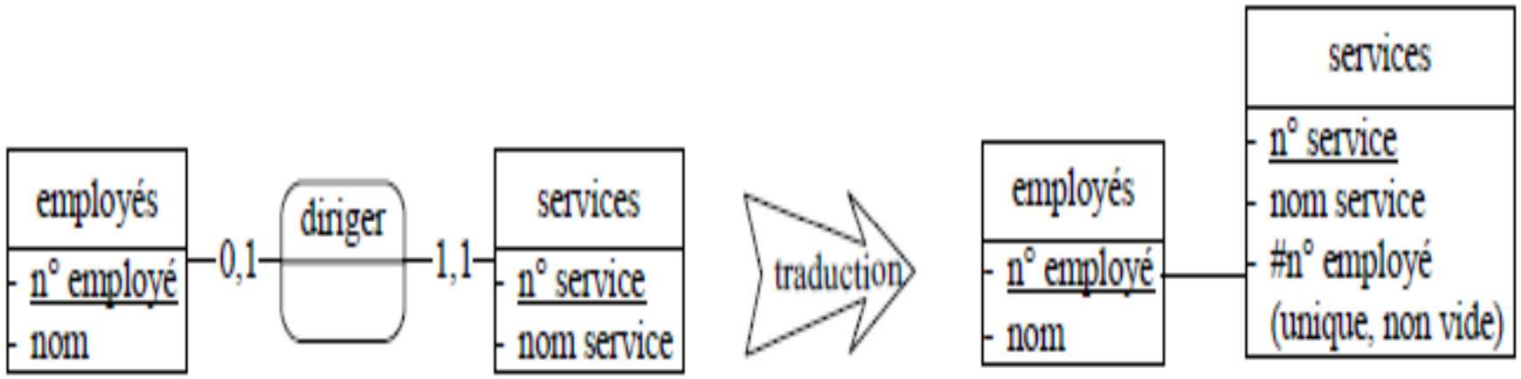


Traduction d'une association de type n : m

Règle 4 : une association binaire de type 1 : 1 est traduite comme une association binaire de type 1 : n sauf que la clé étrangère se voit imposer une contrainte d'unicité en plus d'une éventuelle contrainte de non vacuité (cette contrainte d'unicité impose à la colonne correspondante de ne prendre que des valeurs distinctes).

Si les associations fantômes ont été éliminées, il devrait y avoir au moins un côté de cardinalité 0,1. C'est alors dans la table du côté opposé que doit aller la clé étrangère. Si les deux côtés sont de cardinalité 0,1 alors la clé étrangère peut être placé indifféremment dans l'une des deux tables.

Par exemple, l'association diriger est traduite par :
services(n° service, nom service, #numéro employé (non vide, unique))
employés(numéro employés, nom)

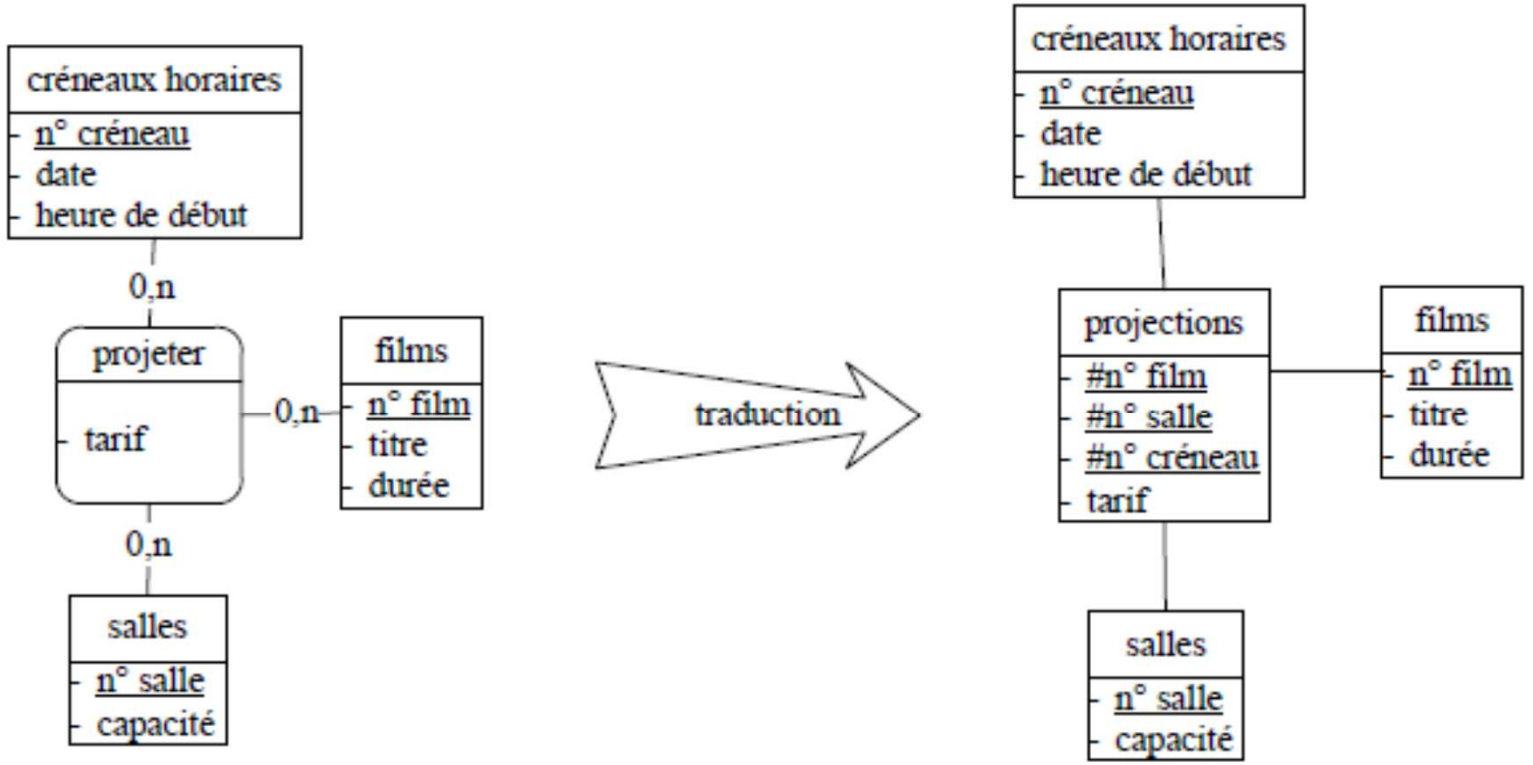


Traduction alternative d'une association de type 1 : 1

Règle 5 : une association non binaire est traduite par une table supplémentaire dont la clé primaire est composée d'autant de clés étrangères que d'entités en association. Les attributs de l'association deviennent des colonnes de cette nouvelle table.

Par exemple, l'association projeter devient la table :

projections(#n° film, #n° salle, #n° créneau, tarif)



Traduction d'une association ternaire

3 Modèle physique de données (MPD)

Un modèle physique de données est l'implémentation particulière du modèle logique de données un logiciel.

3.1 Distinction entre MLD et MPD

La traduction d'un MLD conduit à un MPD qui précise notamment le stockage de chaque donnée à travers son type et sa taille (en octets ou en bits). Cette traduction est également l'occasion d'un certaine nombre de libertés prises par rapport aux règles de normalisation afin d'optimiser les performances du
système d'information.

La traduction d'un MLD relationnel en un modèle physique est la création (par des requêtes SQL de type *CREATE TABLE* et *ADD CONSTRAINT*) d'une base de données hébergée par un SGBD relationnel particulier.
Il peut s'agir d'une base Oracle, d'une base SQL Server, d'une base Access ou d'une base DB2,par exemple. Le fait que tous les SGBDR reposent sur le même modèle logique (le schéma relationnel) permet à la fois la communication entre des bases hétérogènes et la conversion d'une base de donnée d'une SGBDR à l'autre.

3.2 Optimisations

L'optimisation des performances en temps de calcul se fait toujours au détriment de l'espace mémoire consommé. Dans le pire des cas, réduire les temps de réponse consiste à dénormaliser volontairement le système d'information, avec tous les risques d'incohérence et les problèmes de gestion que cela comporte.

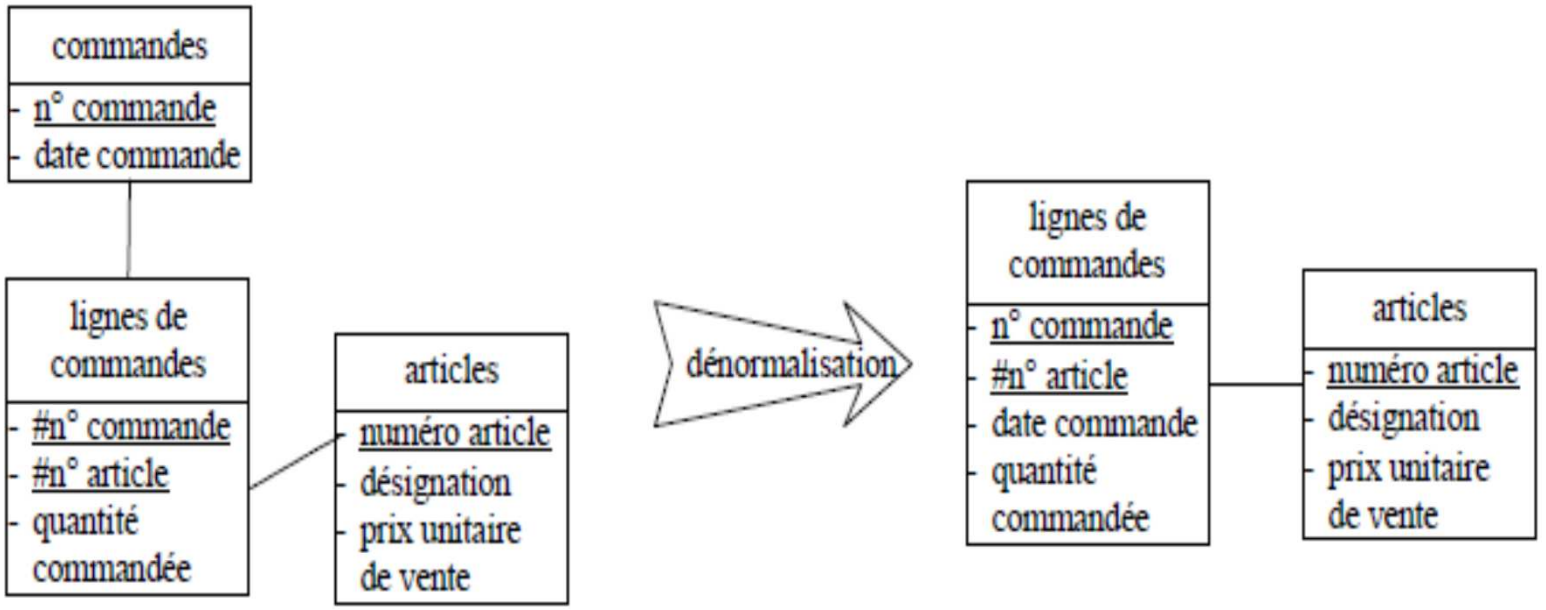
Pour les bases de données relationnelles, l'optimisation qui vise à accélérer les requêtes peut passer par :

- ✓ l'ajout d'index aux tables (au minimum sur les colonnes clés primaires et clés étrangères) ; ces index consomment de l'espace mémoire supplémentaire, mais la base de données reste normalisée ;

- ✓ l'ajout de colonnes calculées ou de certaines redondances pour éviter des jointures coûteuses (auquel cas la base est dé normalisée) ; il faut alors veiller à ce que la cohérence entre les colonnes soit respectée, soit par l'utilisation de déclencheurs, soit dans les applications clientes du système d'information ;

- ✓ la suppression des contraintes d'unicité, de non vacuité ou encore de clé étrangère (auquel cas, l'intégrité des données doit être assurée par le code client du système d'information).

Par exemple, la table *commandes* de la figure 28 peut être supprimée et la date de commande est alors ajoutée à la table *lignes de commandes*. On renonce donc à la troisième forme normale (figure 32)



Sacrifice de la troisième forme normale

puisque la date de commande est répétée autant de fois qu'il y a de lignes dans la commande, mais on évite ainsi une jointure coûteuse en temps de calcul lors des requêtes SQL