UTC504 – Systèmes d'Information et Bases de Données

Conception d'une base de données – MERISE

Sébastien Fourestier

2022

Plan

Plan

Bibliographie

- Ce cours est très largement inspiré de celui de Cyril Gruau : Conception d'une base de données, `http://cyril-gruau.developpez.com/merise/`
- Il est également inspiré par les cours réalisés par des enseignants de l'ENSEIRB-MATMECA : A. Zemmari, M. Mosbah, B. Le Saëc
- Outil de réalisation de diagrammes entité-association : `http://www.analysesi.com/`

Correctifs

- Ce cours est disponible sous licence libre sur ce dépôt github : `https://github.com/sfourestier/enseignement`

→ Voici pouvez :

  - L'améliorer en proposant des *Pull requests*
  - Partager autour de points pouvant être améliorés en créant des tickets (*Issues*)

Historique

- Informatique :
  Construire des systèmes pour effectuer des calculs (équations différentielles, calcul matriciel, etc.)
- Aujourd'hui, on s'appuie de plus en plus sur des données → gestion de grandes quantités d'informations :
  - Stocker des données
  - Manipuler ces données
- Données de natures diverses, opérations plus ou moins compliquées

→ On se place donc dans le cas des systèmes d'informations de gestion
  - Manipulent beaucoup de données

  → Modélisation de la manière dont les données seront organisées

Objectif d'un SGBD

- Stocker, centraliser des données (BD) et les mettre à disposition des utilisateurs
- Manipuler (de manière transparente pour l'utilisateur) des données (SGBD)
- Exemples d'utilisation :
  - Gestion : paye, stock, etc.
  - Transactionnelles : banque, réservation, etc.
  - Documentation : bibliothèque, cartographie, etc.

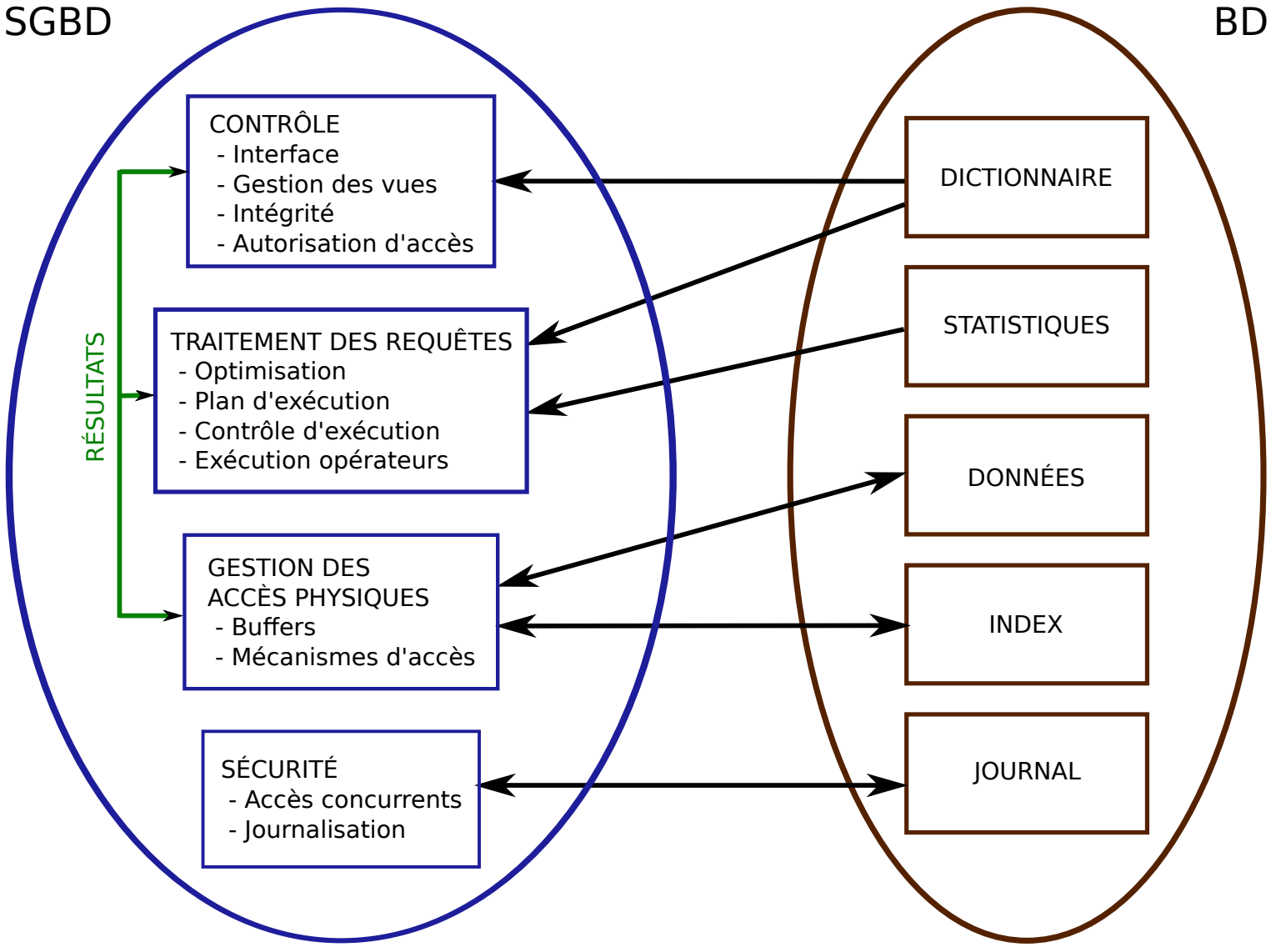
Fonctionnalités d'un SGBD (1)

- Gestion du stockage :
  Tailles importantes des données, éviter les redondances
- Persistance :
  Les données survivent aux programmes qui les créent
- Fiabilité :
  Mécanismes de reprise sur pannes (logiciel ou matériel)
- Sécurité/Confidentialité :
  Contrôle des utilisateurs et des droits d'accès aux données
- Interfaces homme-machine :
  Ergonomie, profils utilisateurs
- Distribution :
  Données stockées sur différents sites
- Optimisation

Fonctionnalités d'un SGBD (2)

- Contrôle de concurrence : propriétés ACID, transactions
  - Atomicité :
    Soit toutes les opérations de la transaction sont validées, soit aucune ne l'est
  - Cohérence :
    Préservation de la cohérence de la base (contraintes d'intégrité, etc.)
  - Isolation :
    Quelle que soit la manière dont les transactions concurrentes sont exécutées, on doit pouvoir les ordonner de sorte à ce que l'état final de la base soit le même qu'après une exécution séquentielle des différentes transactions.
  - Durabilité :
    Si une transaction est validée, tous les changements qu'elle a effectués sur la base sont persistants

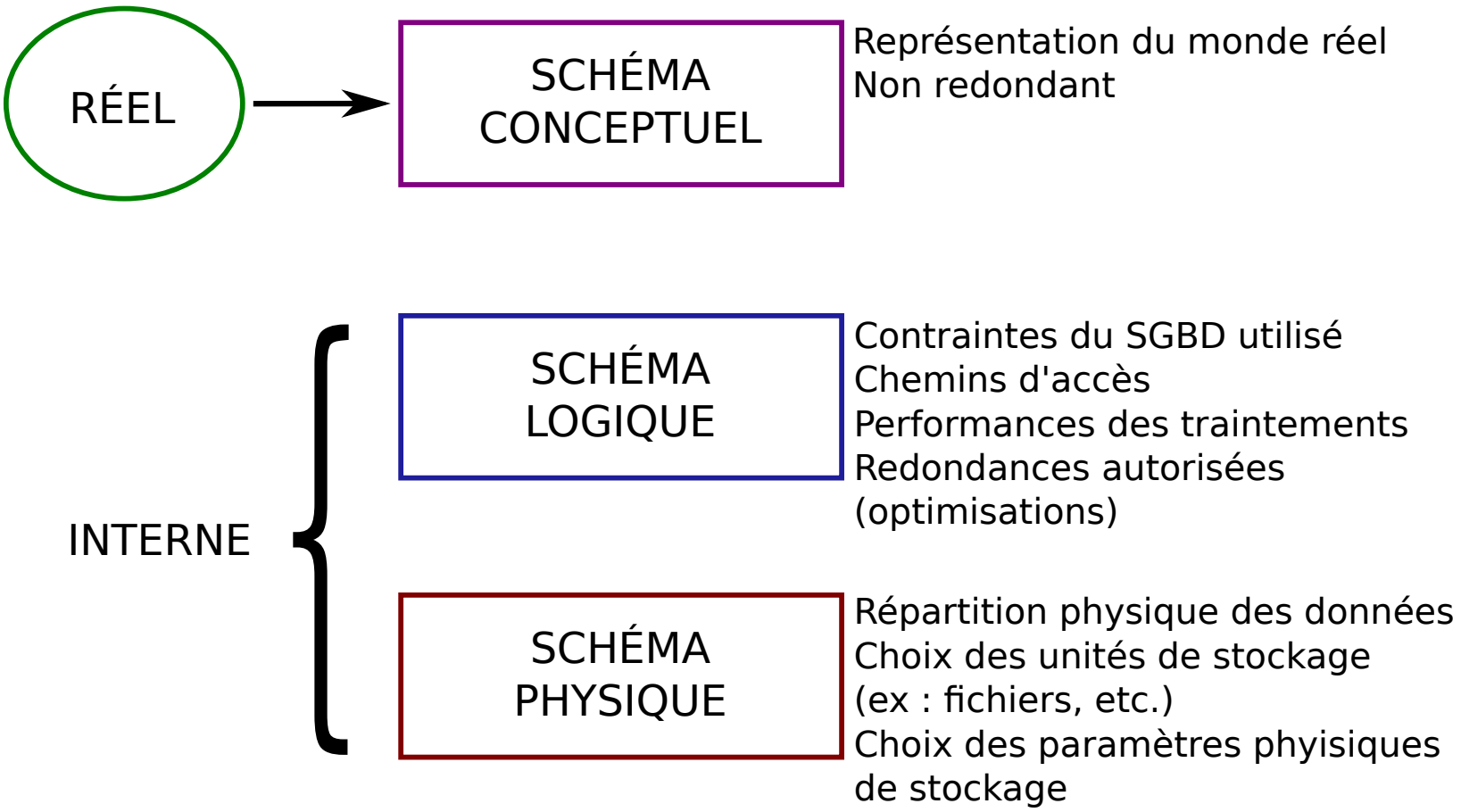
Architecture fonctionnelle d'un SGBD



Utilisateurs d'un SGBD

- Administrateur
  - Définition du schéma logique
  - Définition des structures de stockage et les méthodes d'accès
  - Gestion des autorisations
  - Spécification des contraintes
  - Maintenance de la performance
- Concepteur et programmeur
  - Est informaticien
  - Connaît au moins le LMD
  - Connaît bien le SGBD
  - Connaît un ou plusieurs langages de programmation
- Utilisateur
  - Intervient en amont de la réalisation du SGBD
  - Peut participer à la validation du schéma conceptuel
  - Secrétariat, caissier, etc.

Niveau d'abstraction des données



Modèles de données

On utilise la méthode MERISE :

- Modèle conceptuel de données :
  - Entité-association
  - (Diagramme de classes UML)
- Modèle logique de données :
  - Modèle relationnel
- Modèle physique de données :
  - Implémentation particulière du modèle logique de données par le logiciel

Conception d'une base de données

- Conception d'une base de données :
  1. Analyse des besoins
  2. Description conceptuelle
  3. Conception logique
  4. Conception physique
- Les 2 premières phases sont indépendantes du SGBD
- Le passage de 2 à 3 peut être en partie automatisé

Plan

Entité

- Entité : population d'individus homogènes

- Les produits ou les articles vendus par une entreprise sont de même nature (désignation, prix, etc.)

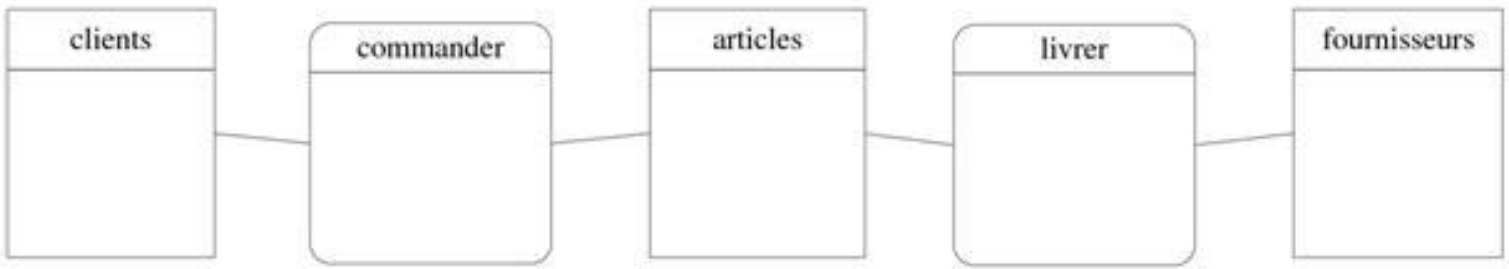
$\rightarrow$ Ils peuvent être regroupés dans une même entité *articles*

- Les *articles* et les *clients* ne sont pas de même nature (informations non homogènes)

$\rightarrow$ On utilise deux entités distinctes

Association

- Association : liaison qui a une signification précise entre plusieurs entités



- L'association *commander* relie les entités *articles* et *clients*
- L'association *livrer* relie les entités *articles* et *fournisseurs*
- L'entité *clients* et relié indirectement à l'entité *fournisseurs* via l'entité *articles*

Attribut

- Attribut : propriété d'une entité ou d'une association

- Le *numéro article* et le *prix unitaire* sont des attributs de l'entité *articles*,
  la *quantité commandée* est un attribut de l'association *commander*, etc.
- Une entité et ses attributs ne doivent traiter que d'un seul sujet

→ Mettre les informations relatives aux fournisseurs dans une entité *fournisseurs* séparée plutôt que dans l'entité *articles*

Identifiant

- Identifiant : attribut sans doublon qui identifie l'entité de manière unique

- Par convention, on souligne l'attribut identifiant sur le schéma
- En général, si il n'y a pas d'attribut adapté, on ajoute un numéro auto-incrémenté
- Une entité possède au moins un attribut (son identifiant)
- Une association peut être dépourvue d'attribut

Cardinalités (1)

- Cardinalité : pour un lien entre une entité et une association, elle précise le minimum et le maximum de fois qu'un individu de l'entité peut être concerné par l'association

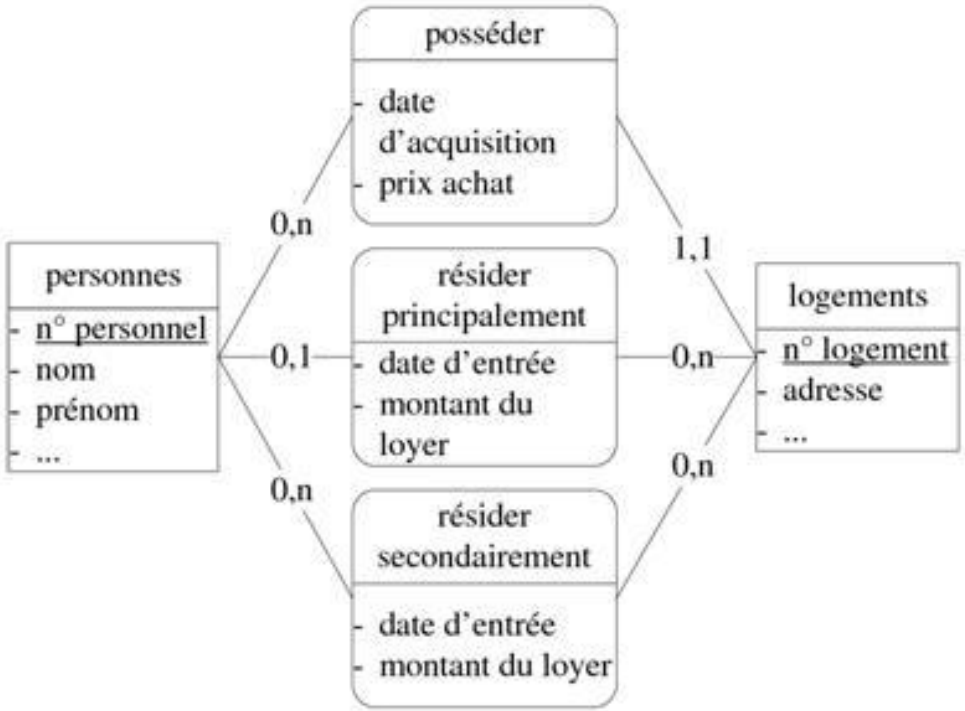
- Exemple :
  - Un *client* a au moins commandé un *article* et peut en commander *n*
  - Un *article* peut avoir été commandé entre 0 et *n* fois

Cardinalités (2)

- Une cardinalité minimale de 0 : les individus de l'entité peuvent exister seuls
- Une cardinalité minimale de 1 : les individus de l'entité ont besoin de l'association pour exister (un client n'existe que si il a commandé)
- Une cardinalité maximale de 1 : les individus de l'entité sont reliés au maximum à un autre individu de l'autre entité
- Une cardinalité maximale de n : les individus de l'entité peuvent reliés à plusieurs individus de l'autre entité
- En général, on n'utilise pas de cardinalités minimales de plus de 1 car elle n'auront pas d'impact par la suite (cf. modèle relationnel)

Associations plurielles

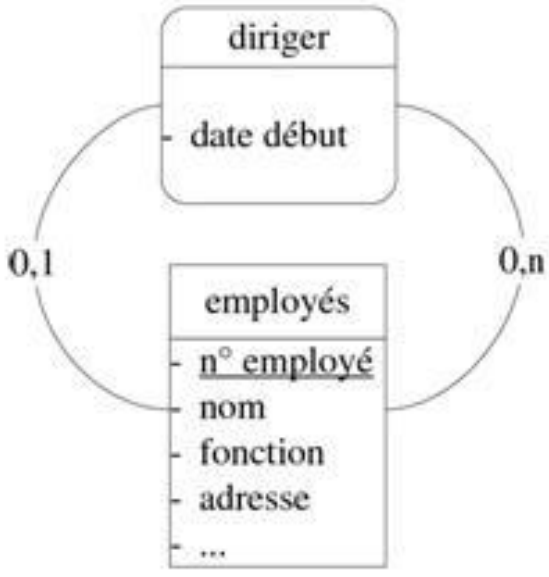
- Deux mêmes entités peuvent être plusieurs fois en association



- Permet d'indiquer des liens de différentes natures

Association réflexive

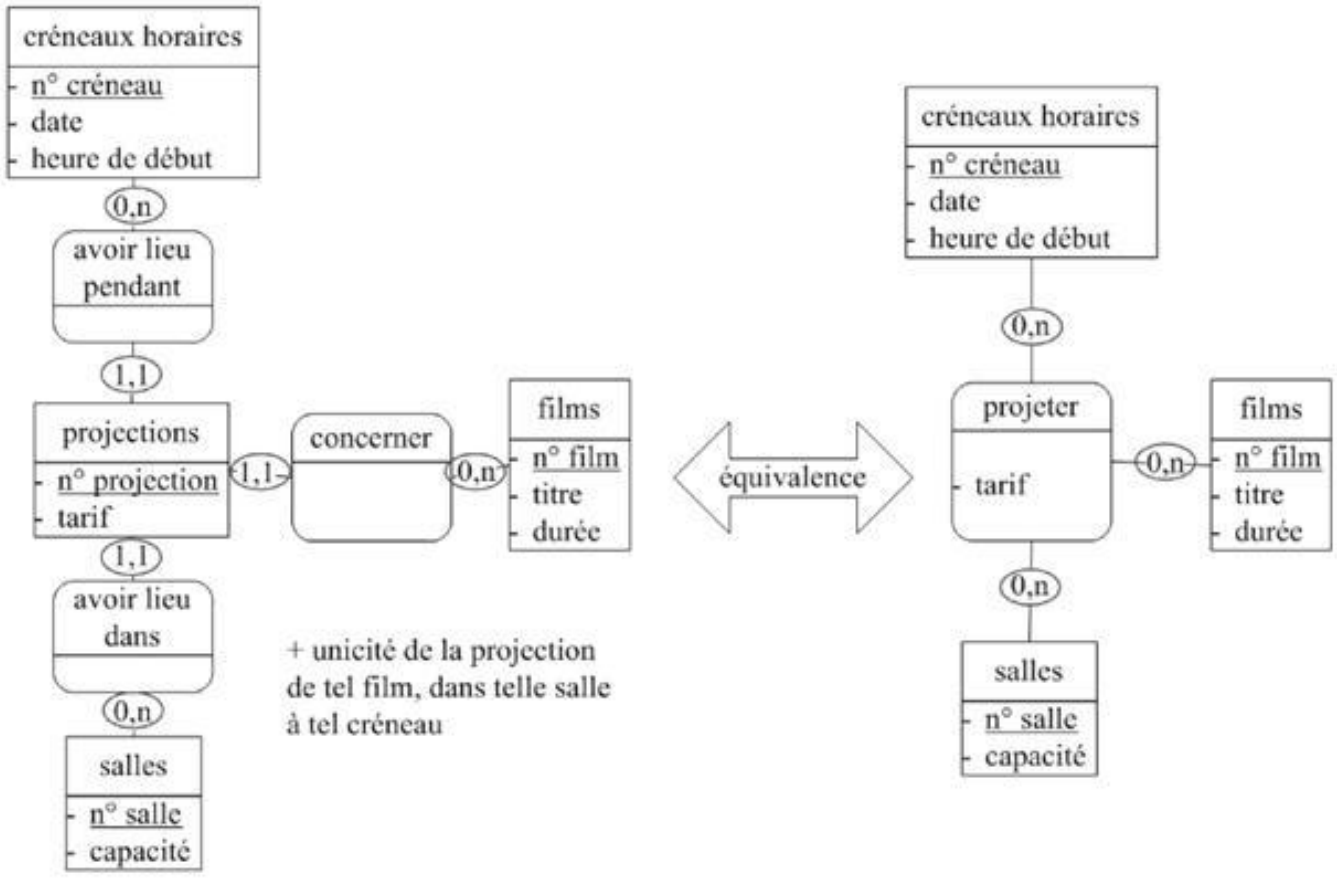
- Une association peut être reliée plusieurs fois avec la même entité



- Un employé est dirigé par un autre employé (sauf le directeur général)
- Un employé peut diriger plusieurs autres employés

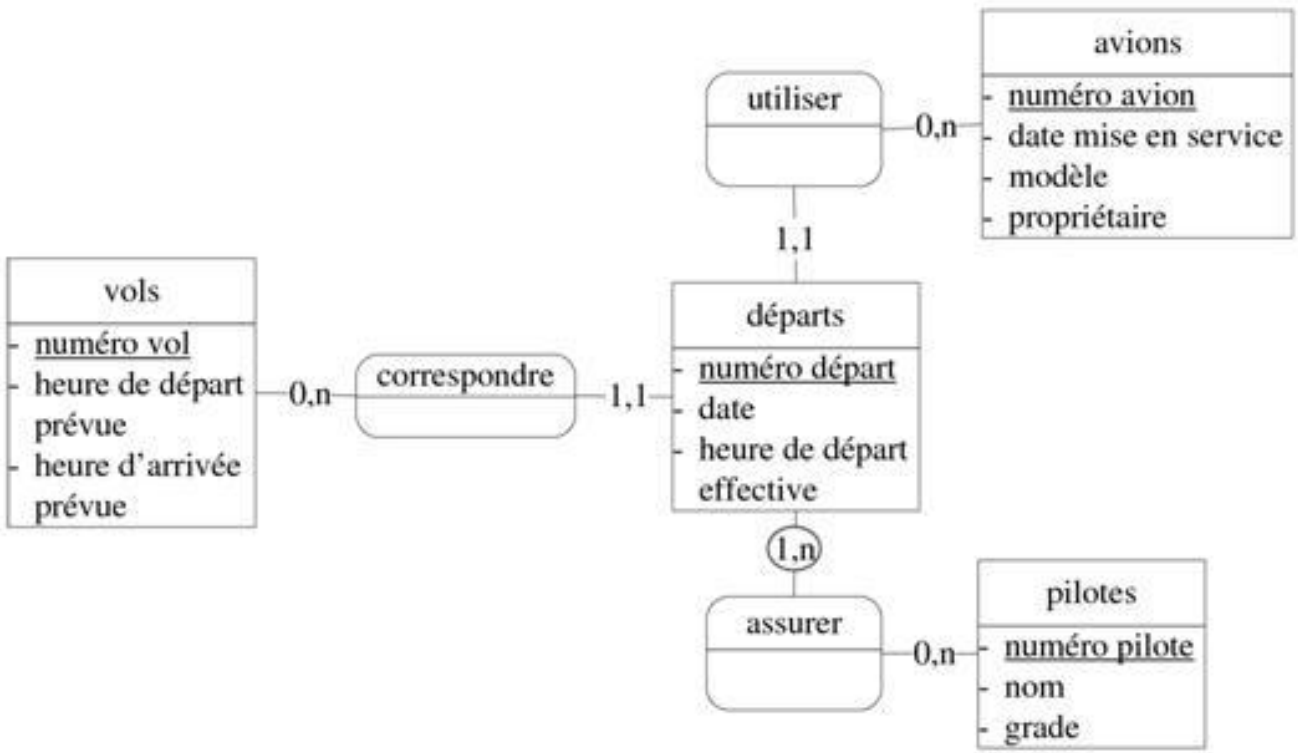
Associations non binaires (1)

- Une entité avec associations de cardinalités maximales 1 au centre et n à l'extérieur peut-être remplaçée par une association avec les cardinalités extérieures



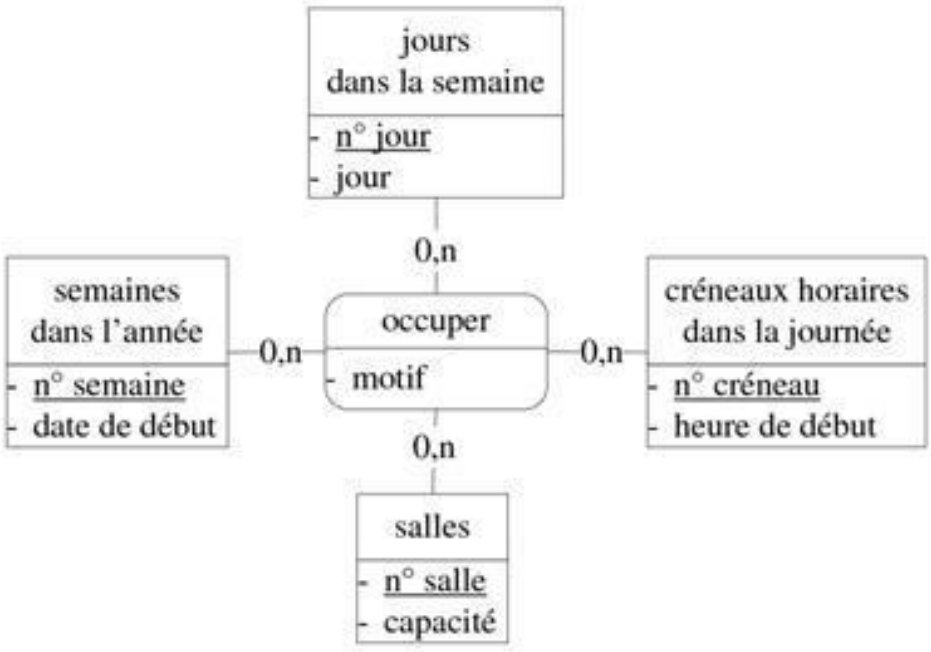
Associations non binaires (2)

- Passer par un schéma entités-associations avec des associations binaires dans un premier temps permet d'éviter les association n-aires abusives (cardinalités non adéquates)



Associations non binaires (3)

- Une association peut être branchée à plus de trois entités



Méthodologie

1. Identifier les entités en présence
2. Lister leurs attributs
3. Ajouter les identifiants
4. Établir les associations binaires entre les entités
5. Lister leurs attributs
6. Calculer les cardinalités
7. Vérifier les règles de normalisation
8. Effectuer les corrections nécessaires

Plan

Bonnes pratiques

- Un bon schéma entités-associations doit répondre à 9 règles de normalisation :
  1. Normalisation des entités
  2. Normalisation des noms
  3. Normalisation des identifiants
  4. Normalisation des attributs
  5. Normalisation des associations
  6. Normalisation des cardinalités
  7. Première forme normale
  8. Deuxième forme normale
  9. Troisième forme normale

Normalisation des entités

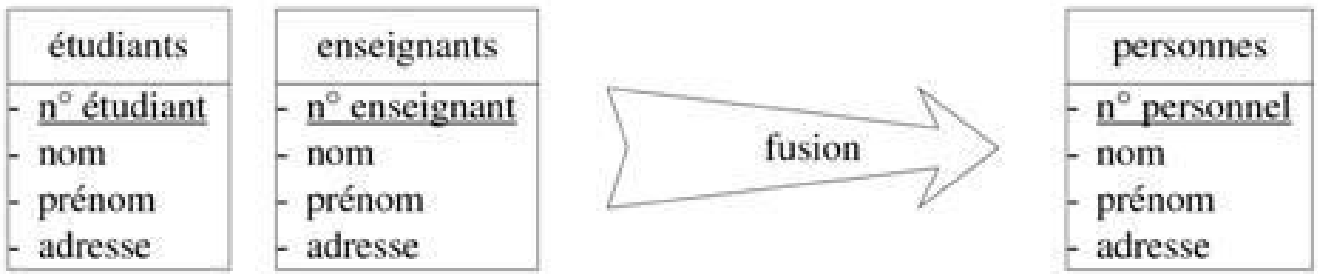
- Toutes les entités qui sont remplaçables par une association doivent être remplacées

Normalisation des noms (1)

- 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 (ex : clients)
  - Pour les associations, utiliser un verbe à l'infinitif (ex : 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 (ex : nom, numéro, libellé, description), éventuellement accompagné du nom de l'entité ou de l'association dans laquelle il se trouve (ex : 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

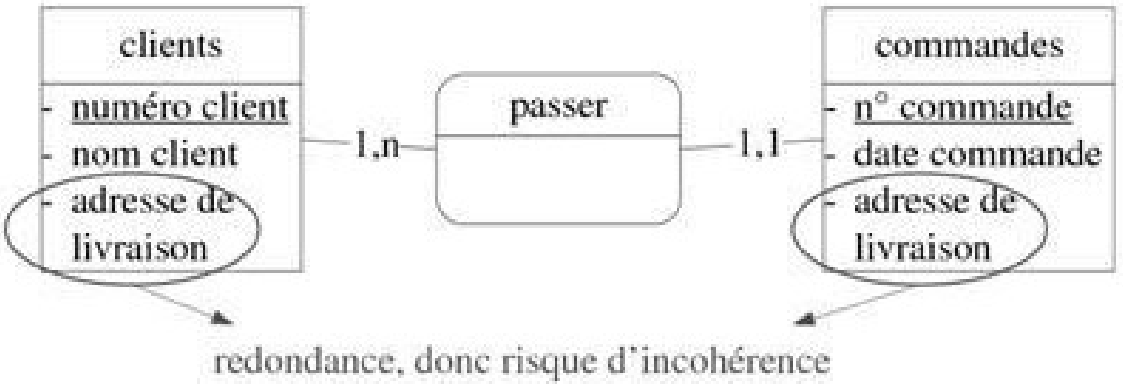
Normalisation des noms (2)

- Deux entités homogènes peuvent être fusionnées



Normalisation des noms (3)

- Éviter les redondances : gaspillage d'espace et risque incohérence



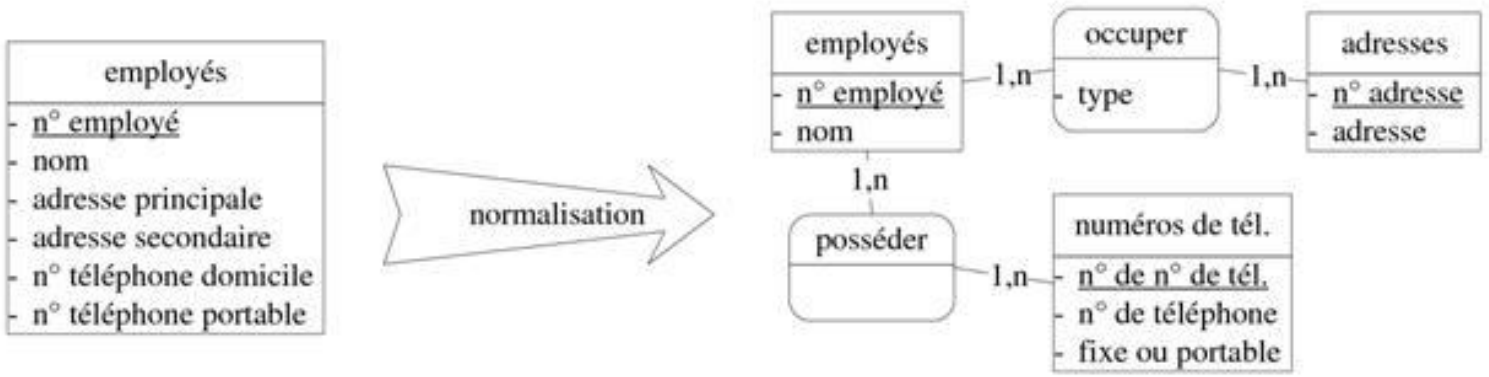
- Si les adresses ne sont pas les mêmes, où faut-il livrer ?

Normalisation des identifiants

- Chaque entité doit posséder un identifiant
- Conseils :
  - Éviter les identifiants composés de plusieurs attributs (ex : nom et prénom) : mauvaises performances et problème d'unicité possible
  - Préférer un identifiant court pour une recherche plus rapide (éviter notamment les chaînes de caractères complexes)
  - Éviter les identifiants susceptibles de changer au cours du temps (ex : les plaques d'immatriculation)
- Conclusion : en général, l'identifiant est un entier, souvent incrémenté automatiquement

Normalisation des attributs (1)

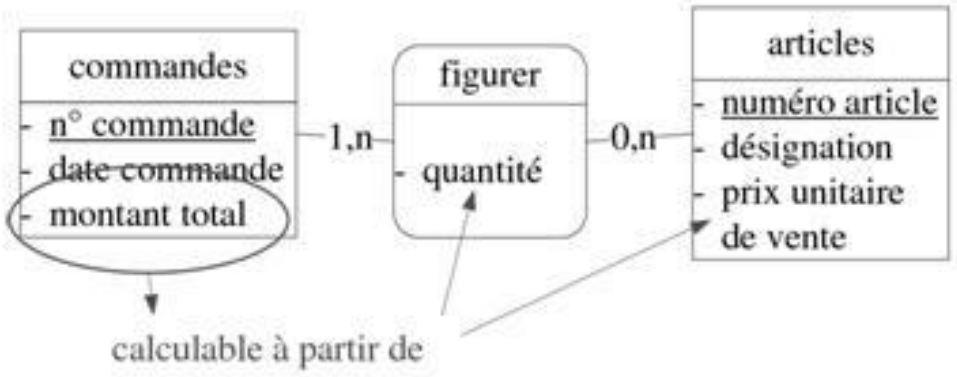
- Remplacer les attributs en plusieurs exemplaires en une association supplémentaire de cardinalités maximales *n*



- Problème d'évolutivité des attributs en plusieurs exemplaires : comment faire si un employé a deux adresses secondaires ?

Normalisation des attributs (2)

- Ne pas avoir d'attribut calculable à partir d'autres attributs



- Risque d'incohérence entre les valeurs des attributs de base et celles des attributs calculés
- Attributs calculables classiques à éviter :
  - l'âge : calculable à partir de la date de naissance
  - le département : calculable à partir du code postal

Normalisation des attributs des associations (1)

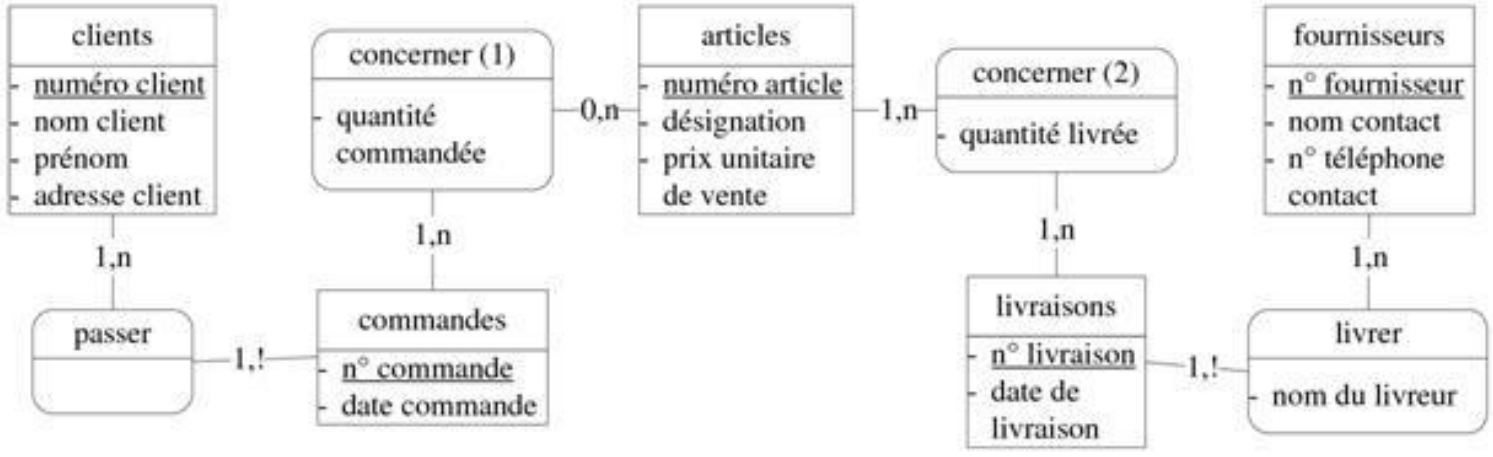
- Les attributs d'une association doivent dépendre directement des identifiants de toutes les entités en association

- La quantité commandée dépend de numéro de client et du numéro d'article, la date de commande non

→ Création d'une entité commandes (idem pour les livraisons)

Normalisation des attributs des associations (2)

- Les attributs d'une association doivent dépendre directement des identifiants de toutes les entités en association

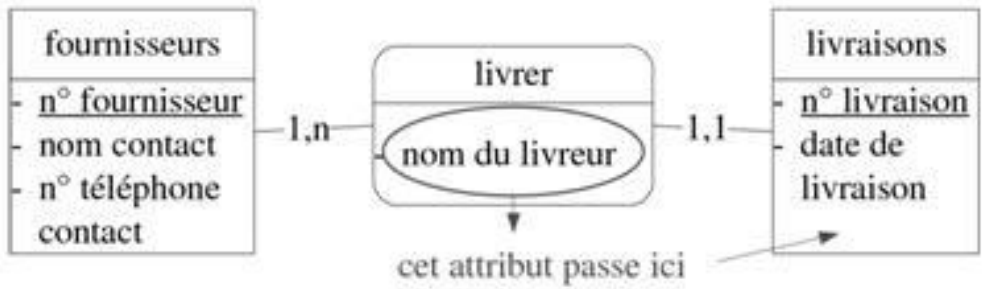


- La quantité commandée dépend de numéro de client et du numéro d'article, la date de commande non

→ Création d'une entité commandes (idem pour les livraisons)

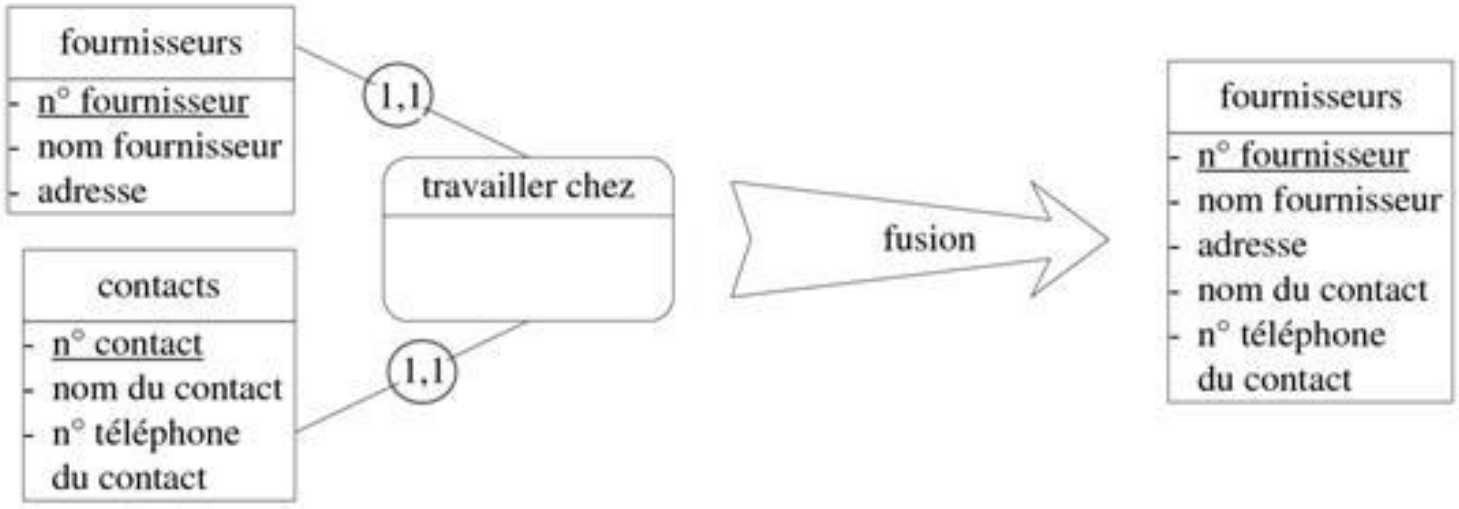
Normalisation des attributs des associations (3)

- Une entité avec une cardinalité de 1,1 ou 0,1 aspire les attributs de l'association



Normalisation des associations (1)

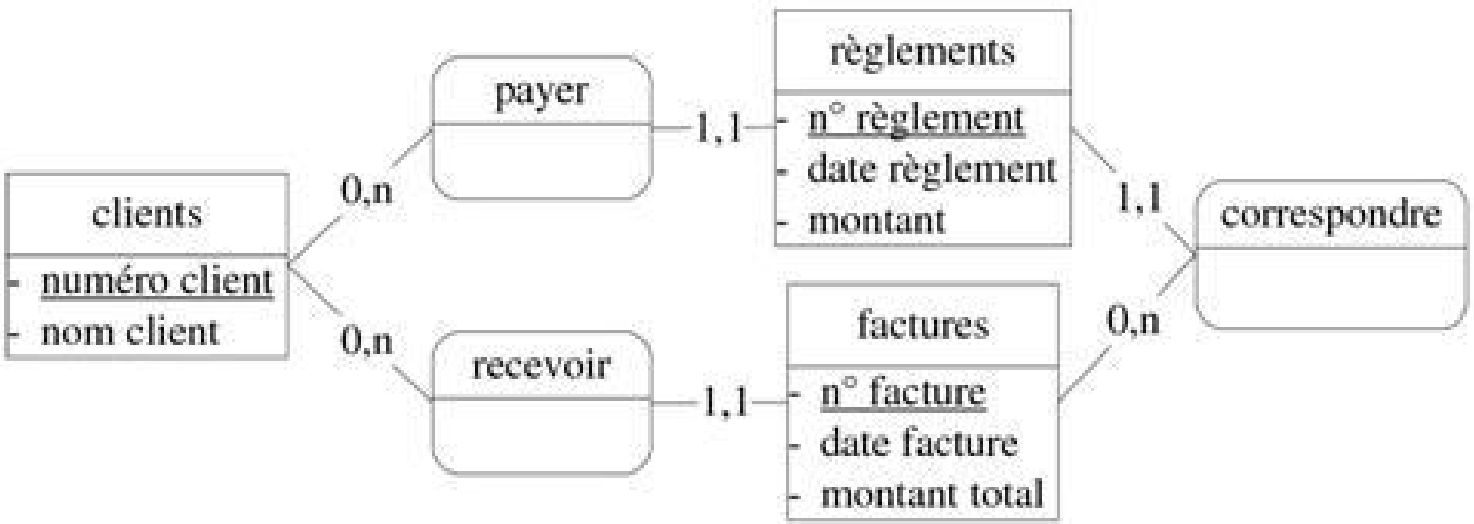
- On élimine les associations fantômes, redondantes ou en plusieurs exemplaires



- Les cardinalités sont toutes 1,1 donc c'est une association fantôme

Normalisation des associations (2)

- Les associations redondantes signifient qu'il existe deux chemins pour aller d'une entité à une autre :
  - Ils doivent avoir deux significations ou deux durées de vie différentes
  - Ou le chemin le plus court doit être supprimé

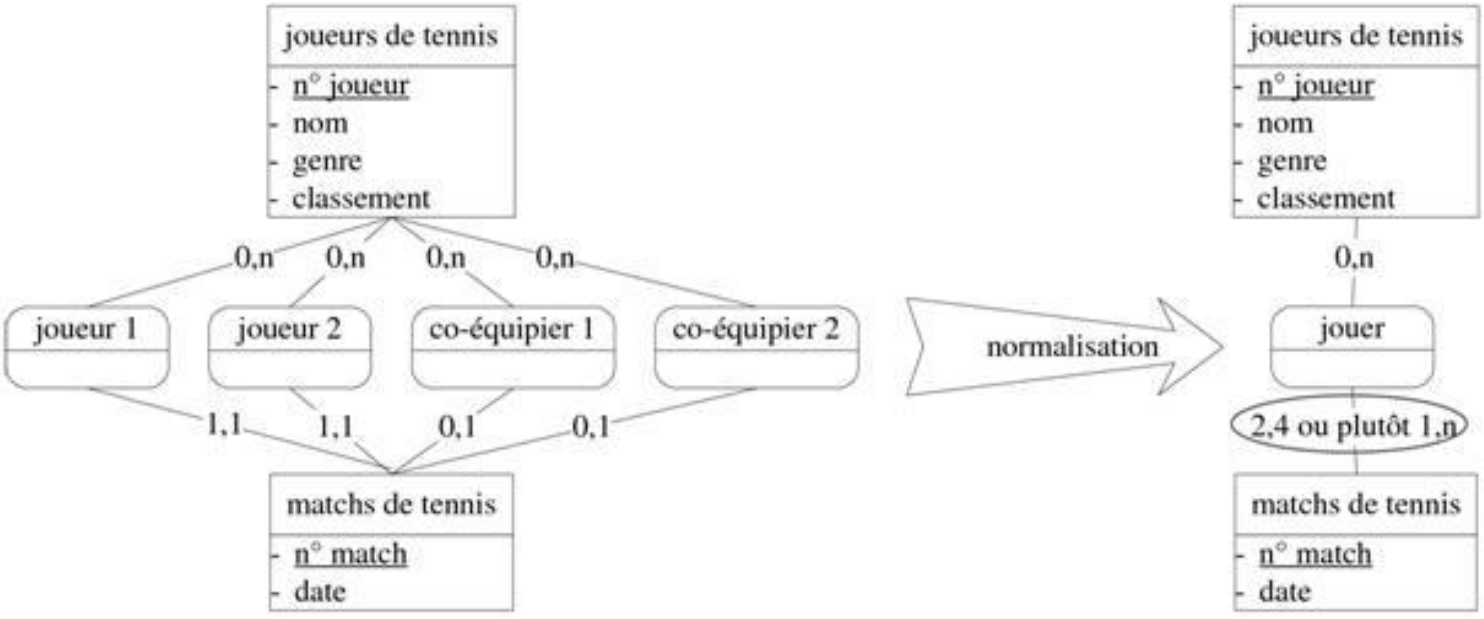


- Si un client ne peut pas régler la facture d'un autre client, alors l'association payer est inutile

→ Elle doit être supprimée

Normalisation des associations (3)

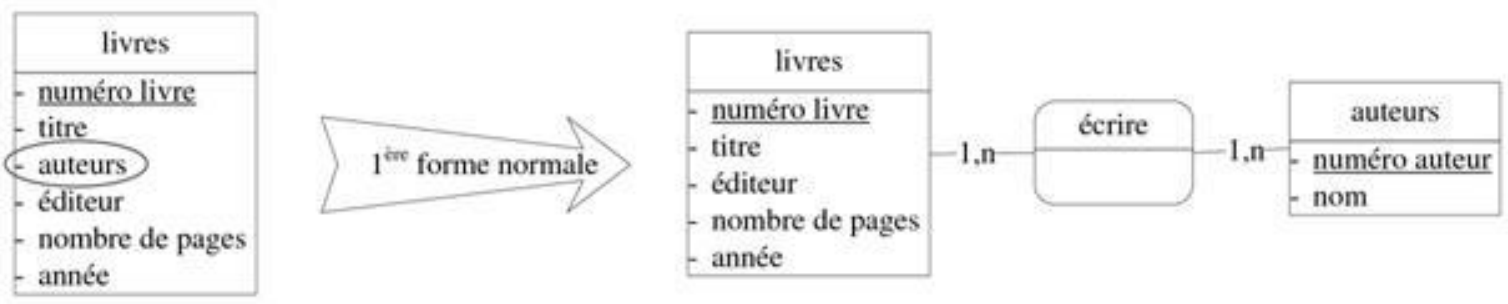
- Les associations en plusieurs exemplaires doivent, si elle le peuvent, être regroupée
  - Évolutivité : on privilégie les cardinalités maximales *n*
  - Une cadinalité minimale de plus de 1 est inutile si la cardinalité maximale est *n*



- Une association suffit pour remplacer les 4 associations

Première forme normale

- « Tous les attributs doivent posséder une valeur sémantiquement atomique. »
- Conséquence : à un instant donné un attribut ne peut prendre qu'une valeur et non pas, un ensemble ou une liste de valeurs



- Si un attribut prend plusieurs valeurs, alors ces valeurs doivent faire l'objet d'une entité supplémentaire, en association avec la première

Deuxième forme normale

- « En première forme normale et :
  Un attribut non identifiant ne dépend pas d'une partie de l'identifiant mais de tout l'identifiant »
- Cela peut arriver si l'identifiant est composé de plusieurs attributs

$\rightarrow$ Peut être oubliée si l'on utilise des identifiants non composés et de type entier

Troisième forme normale

- « En deuxième forme normale et :
  Tous les attributs non identifiants doivent dépendre directement de l'identifiant »
- En pratique on utilise la troisième forme normale de Boyce-Codd, plus complète

Troisième forme normale de Boyce-Codd (1)

- « En deuxième forme normale et :
  - Tous les attributs non identifiants doivent dépendre directement de l'identifiant
  - Aucune partie de l'identifiant dépend d'un attribut non-identifiant »

Troisième forme normale de Boyce-Codd (2)

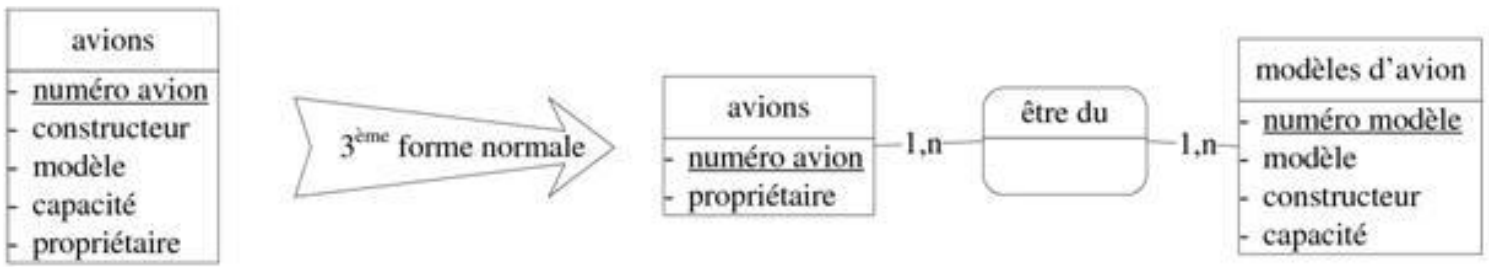
- Version simplifié : tous les attributs d'une entité doivent dépendre directement de son identifiant et d'aucun autre attribut
- Si ce n'est pas le cas, il faut placer l'attribut pathologique dans une entité séparée en association avec la première

numéro avion	constructeur	modèle	capacité	propriétaire
1	Airbus	380	180	Air France
2	Boeing	747	314	British Airways
3	Airbus	380	180	KLM

- Redondance ($\rightarrow$ risque d'incohérence) pour les colonnes constructeur et capacité qui dépendent de modèle

Troisième forme normale de Boyce-Codd (3)

- Version simplifié : tous les attributs d'une entité doivent dépendre directement de son identifiant et d'aucun autre attribut
- Si ce n'est pas le cas, il faut placer l'attribut pathologique dans une entité séparée en association avec la première



- La colonne constructeur devrait également être dans une entité séparée constructeurs (en association avec modèles d'avion)

Plan

Dépendances fonctionnelles

- Méthode pour établir efficacement un modèle entités-associations bien normalisé :
  1. Étudier les dépendances fonctionnelles
  2. Obtention du graphe de couverture minimale
  3. Traduction du graphe en modèle
- Traditionnellement employé pour normaliser des modèles relationnels

Définition

- Un attribut B dépend fonctionnellement d'un attribut A si et seulement si une valeur de A induit une unique valeur de B
- On note une dépendance fonctionnelle par une flèche simple :

$$A \rightarrow B$$

Transitivité

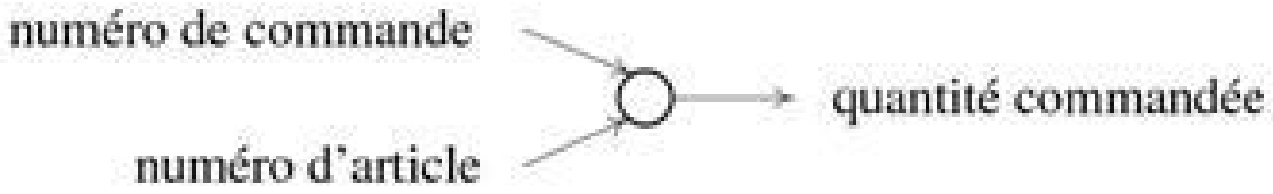
- Une dépendance fonctionnelle est transitive :

$$\text{si } A \to B \text{ et } B \to C \text{ alors } A \to C$$

- Exemple :
  si *numéro commande* $\to$ *numéro client* $\to$ *nom client*
  alors *numéro commande* $\to$ *nom client*
- Deux types de dépendances fonctionnelles :
  - directe : *numéro commande* $\to$ *numéro client*
  - transitive : *numéro commande* $\to$ *nom client*

Dépendances fonctionnelles non élémentaires

- Dépendance fonctionnelles non élémentaire :
  Un attribut B a une dépendance fonctionnelle qui repose sur la conjonction de plusieurs attributs
- On note une dépendance fonctionnelle non élémentaire par une flèche unique avec plusieurs points d'entrée (regroupés autour d'un cercle)



- Dans l'exemple, dépendance fonctionnelle à la fois non élémentaire et directe

Graphe de couverture minimale

- Graphe de couverture minimale :
  Réseau obtenu en représentant tous les attributs et toutes les dépendances fonctionnelles directes entre ces derniers

Clé

Soit une relation R avec $A_1, \ldots, A_n$ attributs.
Soit X un sous-ensemble d'attributs de R, X est une clé ou un identifiant si :

- Pour tous les attributs A_i de R, $X \rightarrow A_i$
- X est le plus petit élément qui détermine tous les autres, c'est-à-dire, il n'existe pas Y un sous-ensemble X avec pour les tous attributs de A_i de R, $X \rightarrow A_i$

Méthodologies

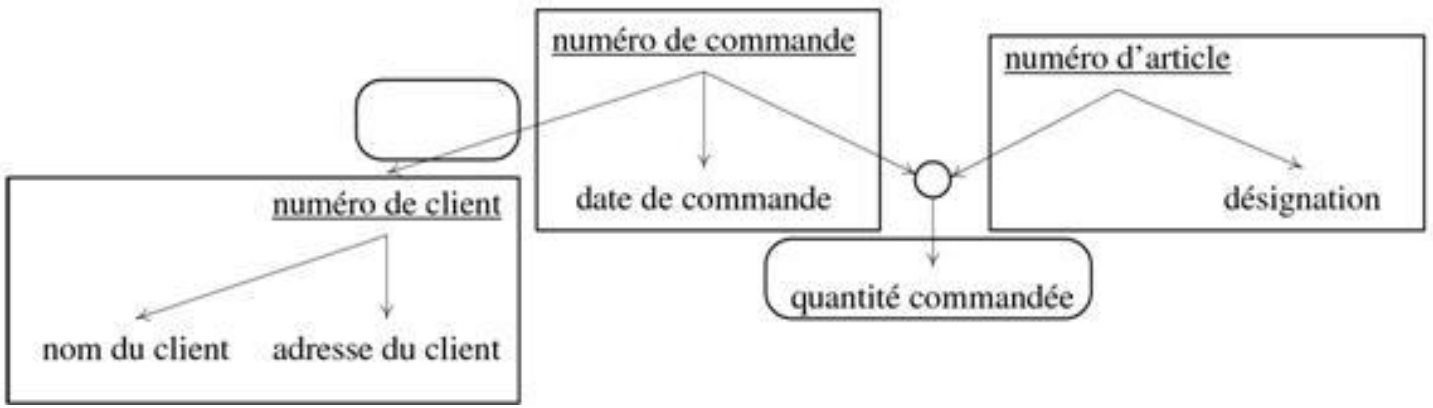
- Nous avons vu à la fin de la section « Modèle conceptuel » une méthodologie pour obtenir un modèle conceptuel de donnée
- Nous allons maintenant voir une méthodologie à partir de l'étude des dépendances fonctionnelles directes

Méthodologie classique

1. Identifier les entités en présence
2. Lister leurs attributs
3. Ajouter les identifiants
4. Établir les associations binaires entre les entités
5. Lister leurs attributs
6. Calculer les cardinalités
7. Vérifier les règles de normalisation, en particulier :
   - Normalisation des entités (associations non binaires, etc.)
   - Normalisation des associations et des attributs
   - Troisième forme normale de Boyce-Codd
8. Effectuer les corrections nécessaires

Traduction à partir des dépendances fonctionnelles

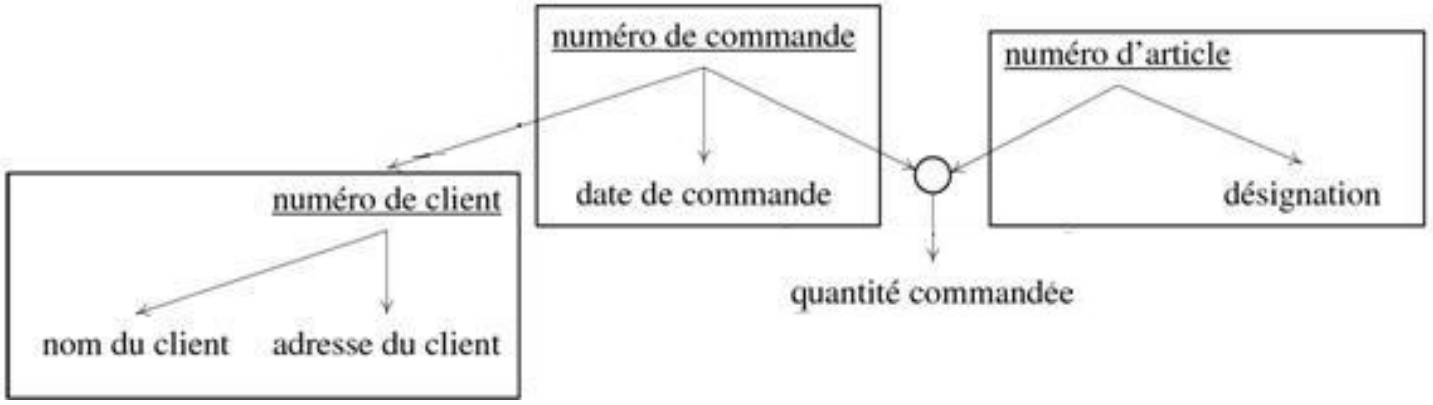
- Plusieurs étapes pour passer du graphe de couverture minimale à un schéma entités-associations normalisé



Étapes de traduction (1)

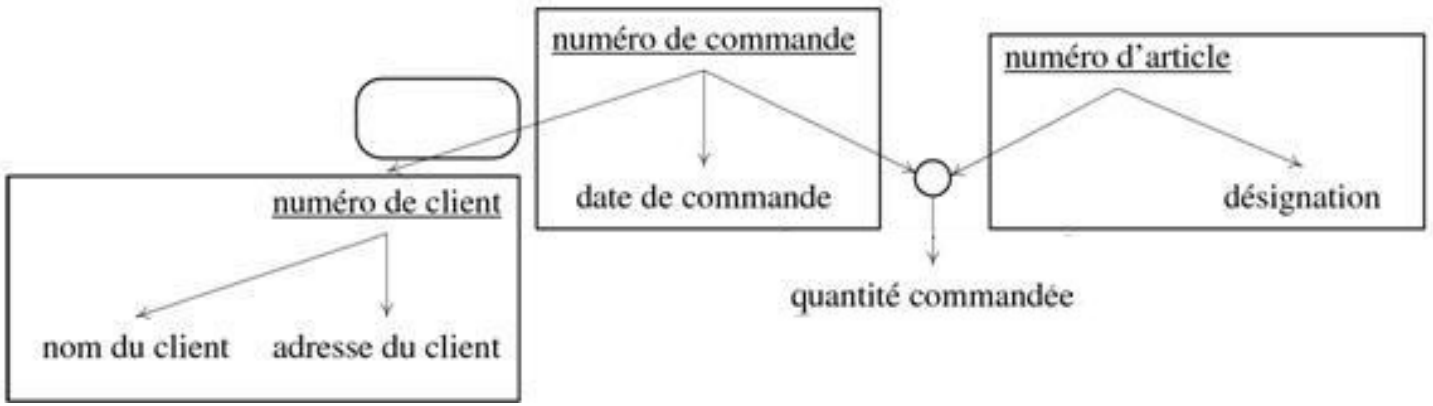
- Étape 1 : repérer et souligner les identifiants (choisir les clés)

Étapes de traduction (2)



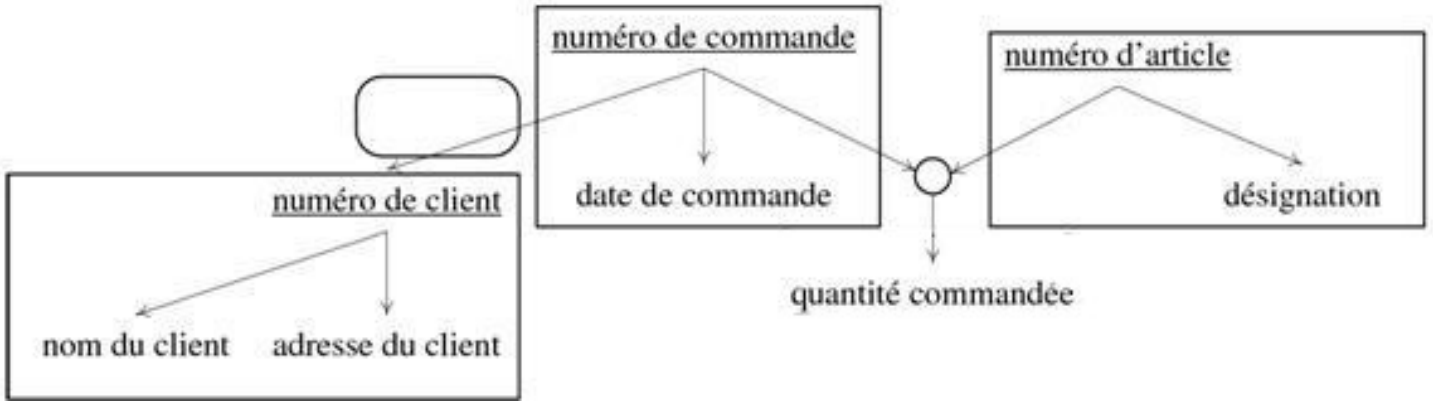
- Étape 2 : tous les attributs non identifiant qui dépendent directement d'un identifiant et d'un seul, forment (avec l'identifiant) une entité

Étapes de traduction (3)



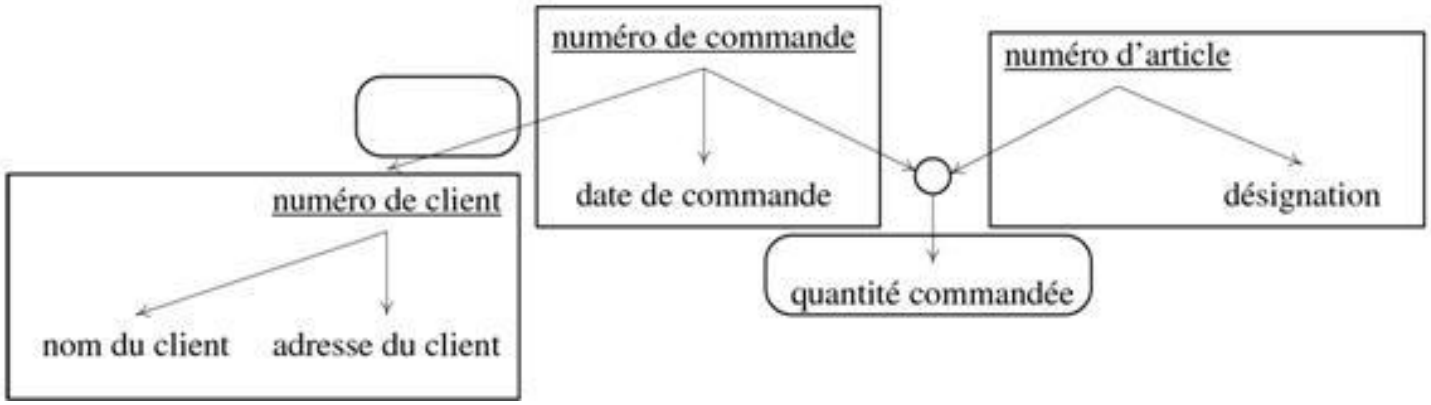
- Étape 3 : les dépendances élémentaires entre les identifiants forment des associations binaires dont les cardinalités maximales sont 1 au départ de la dépendance fonctionnelle et *n* à l'arrivée

Étapes de traduction (4)



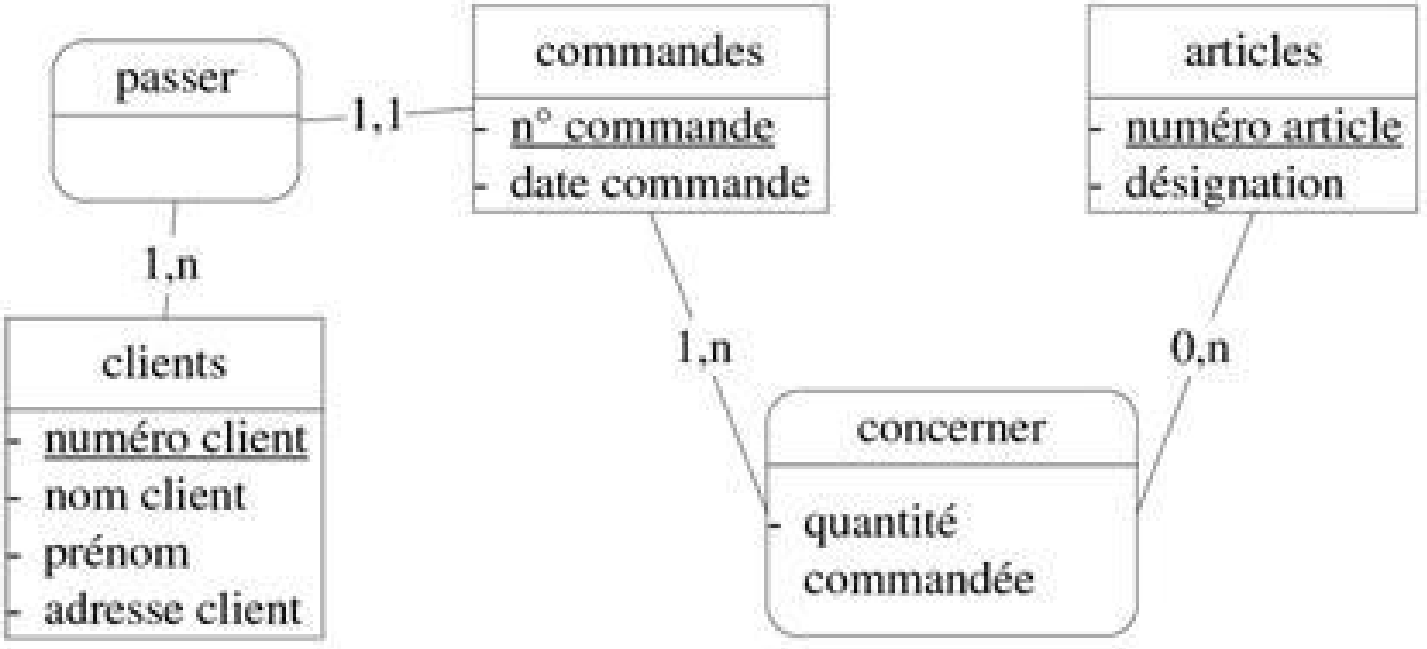
- Étape 4 : sauf si entre deux identifiants se trouvent deux dépendances élémentaires réflexives, auquel cas l'association binaire a deux cardinalités maximales valant 1

Étapes de traduction (5)



- Étape 5 : les attributs (non identifiants) qui dépendent de plusieurs identifiants sont les attributs d'une association supplémentaire dont les cardinalités maximales sont toutes *n*

Schéma obtenu



Remarques

- Il faut donner un nom aux entités et aux associations
- Il reste les cardinalités minimales à établir
- En pratique, il faut connaître les entités pour établir le graphe de couverture minimale

$\rightarrow$ Aide surtout à :

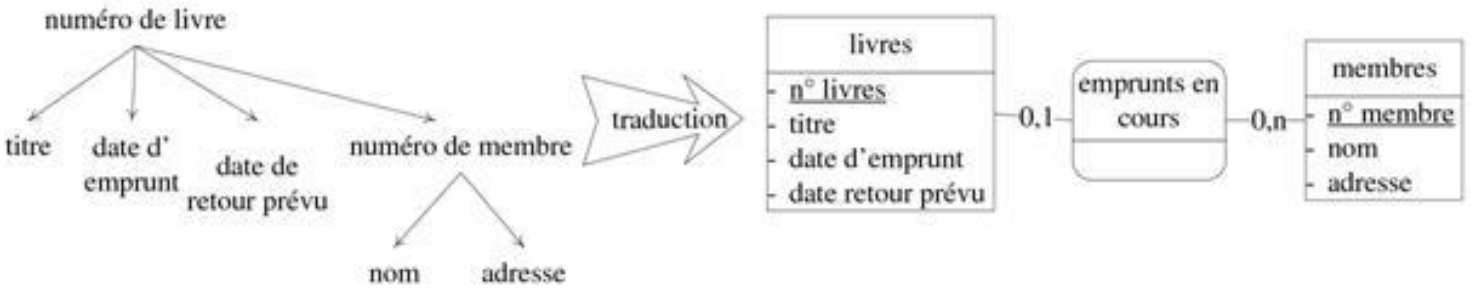
- Établir les associations entre les entités
- Normaliser les entités et leurs associations jusqu'en troisième forme normale de Boyce-Codd

Méthodologie avec dépendances fonctionnelles

- Identifier les entités et leur donner un identifiant
- Ajouter les attributs et leur dépendances fonctionnelles directes avec les identifiants (commencer par les dépendances élémentaires)
- Traduire le graphe de couverture minimale obtenu en un schéma entités-associations
- Ajuster les cardinalités minimales
- La majorité des règles de normalisation devraient être vérifiées, vérifier notamment :
  - La normalisation des noms
  - Les attributs en plusieurs exemplaires
  - Les associations redondantes ou en plusieurs exemplaires

Gestion des dates et de l'historique (1)

- Dans une bibliothèque, on peut vouloir stocker les emprunts en cours ou les emprunts historiques



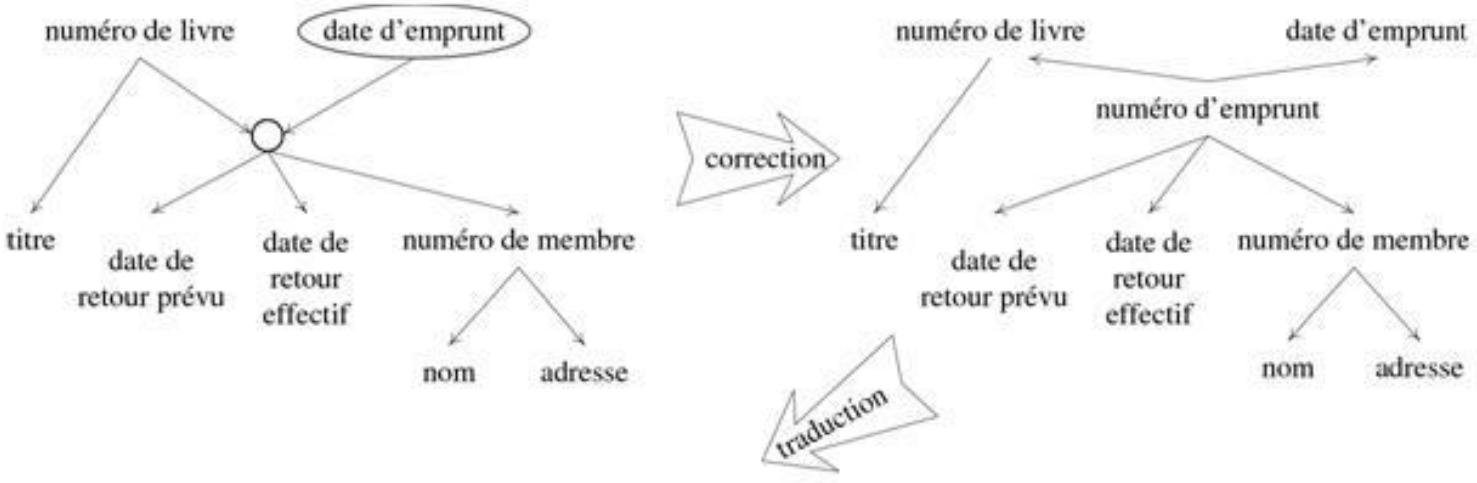
- Emprunts en cours : la date de retour prévu est un attribut de l'entité livres car un livre ne peut faire l'objet que d'un seul emprunt en cours

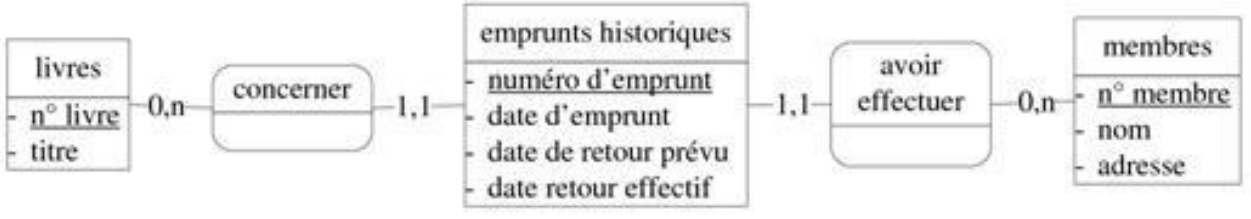
Gestion des dates et de l'historique (2)

- Emprunts historiques :
  - Un livre peut faire l'objet de plusieurs emprunts historiques
  - Date d'emprunt est nécessaire pour connaître la date de retour prévue
  - On évite d'avoir une date comme identifiant
  - Une dépendance fonctionnelle ne peut partir que d'un ou plusieurs identifiant(s)

$\rightarrow$ C'est le signe qu'il manque un identifiant : le numéro d'emprunt

Gestion des dates et de l'historique (3)





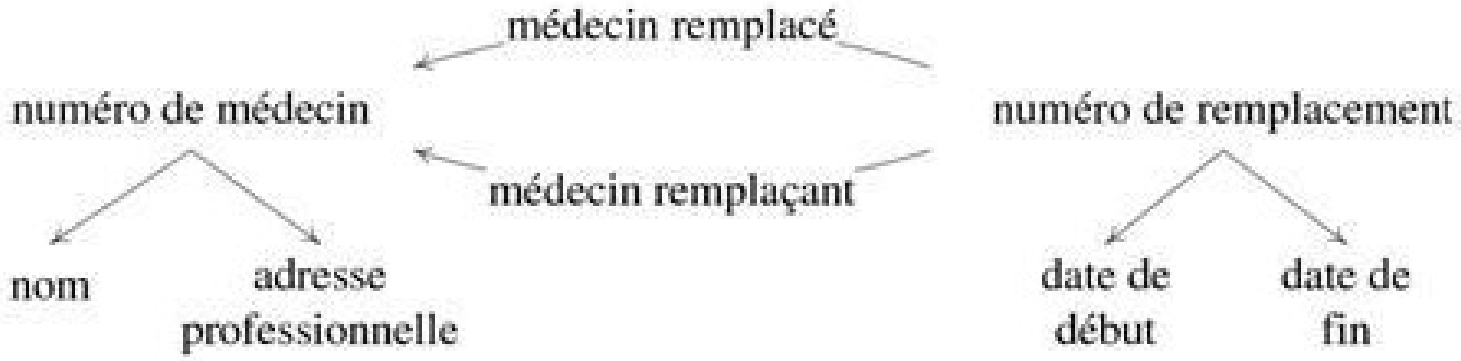
Gestion des dates et de l'historique (4)

- Même pour une entité historisée, il vaut mieux éviter que la date n'entre dans l'identifiant
- Ici, l'entité emprunts historiques ne peut pas être transformée en une association (normalisation des attributs des associations) :
  date retour effectif ne dépend pas du numéro de livre et du numéro de membre, mais du numéro de livre et de la date d'emprunt

$\rightarrow$ Si il n'y avait que le numéro et la date d'emprunt, on pourrait utiliser une association

Dépendances plurielles

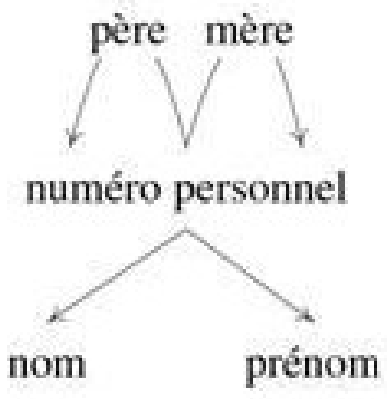
- Une ou plusieurs dépendances fonctionnelles partent ou arrivent plusieurs fois du même attribut



- Ajouter un commentaire sur la flèche permet de clarifier la signification de chaque dépendance
- Ce commentaire donnera le nom des associations correspondantes
- Dépendances fonctionnelles entre médecins et remplacements $\rightarrow$ associations plurielles entre entités médecins et remplacements

Dépendances réflexives

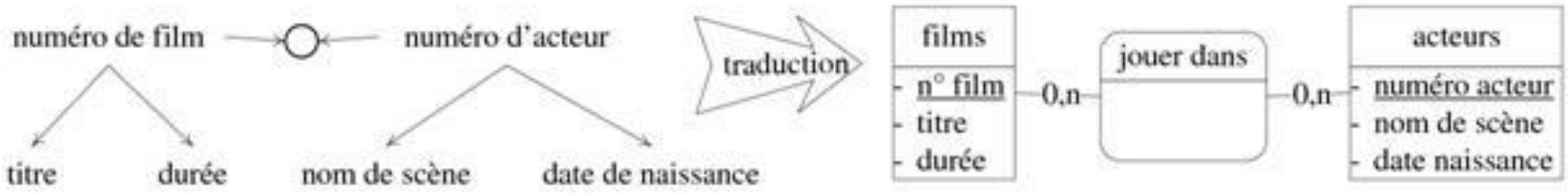
- Dépendances fonctionnelles réflexives : $X \rightarrow X$



- Intérêt : seulement si elles ont une signification particulière

Associations sans attributs

- Attention : associations avec cardinalités maximales *n* et sans attribut ne figurent pas sur le graphe de couverture minimale



- Possiblilité d'introduire une notation spéciale
- Exemple : une dépendance non élémentaire qui ne débouche sur aucun attribut

Plan

Systèmes logiques : historique

- Fichiers binaires et gérées par des programmes exécutables : modification de la structure des données très problématique
- SGBD hiérarchiques : Données organisées en arbre
- SGBD réseaux : Données organisées en graphe
- SGBD hiérarchiques et réseaux : dit navigationnels car on peut retrouver l'information à partir du chemin d'accès
- Pour chacun de ces systèmes, il existe des méthodes de traductions du MLD vers le MPD

Systèmes logiques : historique (2)

- SGBD relationnels (SGBDR) : Information obtenue avec langage quasiment naturel (SQL pour Structured Query Langage)
- SGBD orientés objets : Il existe des *framework* avec des *mapping* objet-relationnel (*ORM*)
- Les bases de données NoSQL
  - Relachement des propriétés ACID
  - Permet la gestion d'un très grand nombre de données (*Big Data*)
- Pour ce cours : modèle logique relationnel, abrégé : « modèle relationnel »

Modèle logique de données

- Le modèle logique de données propose une modélisation plus concrète
- Il est proche du modèle physique de données (qui sera implémenté)
- Il peut être traduit directement depuis le modèle conceptuel de données

Tables, lignes et colonnes

- Table : regroupe les données qui ont la même structure
- Colonne : décrit les champs en commun
- Ligne : contient les valeurs de ces champs pour un enregistrement
- Les lignes d'une table doivent être uniques

numéro client	nom	prénom	adresse
1	Dupont	Michel	127, rue...
2	Durand	Jean	314, boulevard...
3	Dubois	Claire	51, avenue...
4	Dupuis	Marie	2, impasse...
...	...	...	...

- Exemple : table des renseignements clients

Clés primaires

- Clé primaire :
  - Ensemble des colonnes permettant d'identifier une ligne
- Une clé primaire ne peut avoir la valeur vide (NULL)
- La valeur de la clé primaire ne devrait pas changer au cours du temps

Clés étrangères

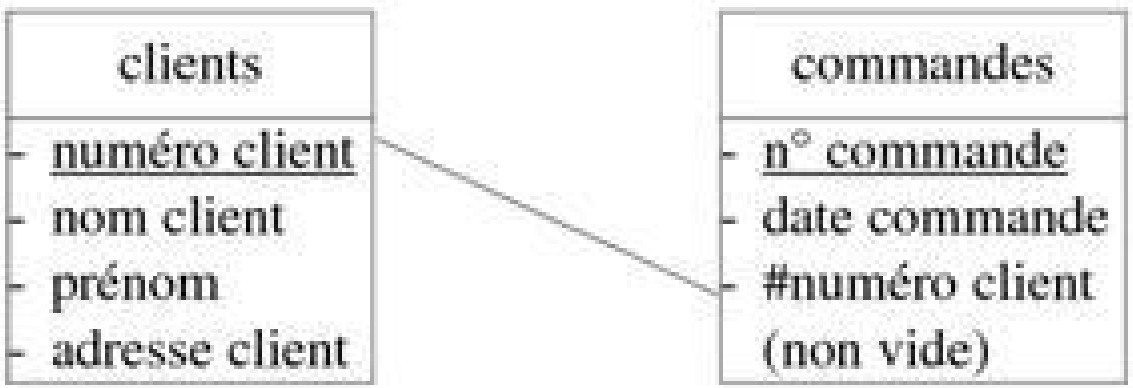
- Une colonne Colonne1 d'une table est clé étrangère ou référence la colonne Colonne2, si :
  - Si elle ne contient que des valeurs prises par la colonne Colonne2 d'une autre table
  - Colonne2 est sans doublons
- Par convention, on souligne les clés primaires et on fait précéder les clés étrangères 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
- Une 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)
- Chaque colonne qui compose une clé primaire ne peut pas recevoir la valeur vide (NULL)

Schéma relationnel

- Représentation des tables d'une base de données relationnelle par un schéma relationnel :
  - Tables sont appelées *relations*
  - Liens entre les clés étrangères et leur clé primaire symbolisés par un connecteur



Traduction d'un MCD en un MLDR

- Pour traduire un MCD en un MLDR, il suffit d'appliquer cinq règles :
  1. Traduction d'une entité
  2. Traduction d'une association binaire 1 : n
  3. Traduction d'une association binaire n : m
  4. Traduction d'une association binaire 1 : 1
  5. Traduction d'une association non binaire

Notations

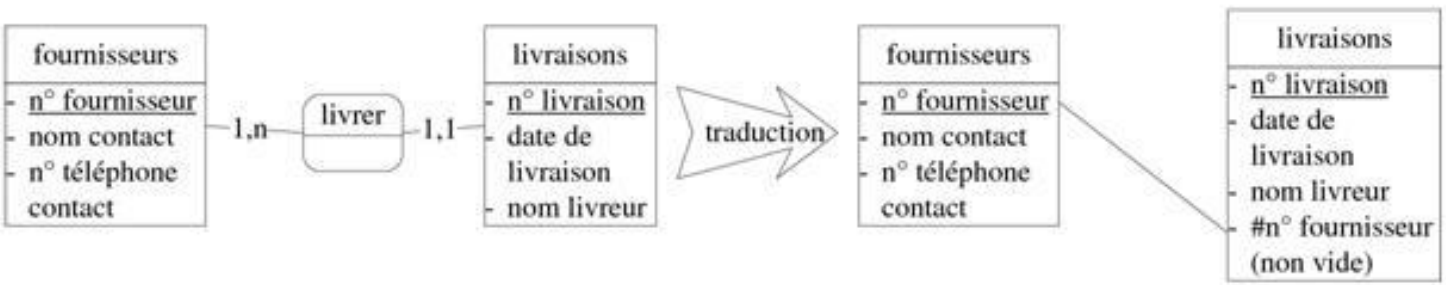
- Une association binaire est de type :
  - $1:1$ (un à un) : si aucune des deux cardinalités maximales n'est n
  - $1:n$ (un à plusieurs) : si une des deux cardinalités maximales est n
  - $n:m$ (plusieurs à plusieurs) : si les deux cardinalités maximales sont n
- Un schéma relationnel ne peut faire la différence entre $0,n$ et $1,n$
- Par contre, il peut la faire entre $0,1$ et $1,1$

Règle 1 : Entité

- Une entité devient une table :
  - Les attributs deviennent les colonnes
  - L'identifiant constitue la clé primaire de la table
- Exemple :
  L'entité article devient :
  `articles(numéro article, désignation, prix unitaire de vente)`

Règle 2 : Association binaire 1 : *n* (1)

- `fournisseurs(numero fournisseur, nom contact, numero téléphone contact)`
- `livraisons(numero livraison, date de livraison, nom livreur, #numero fournisseur (non vide))`

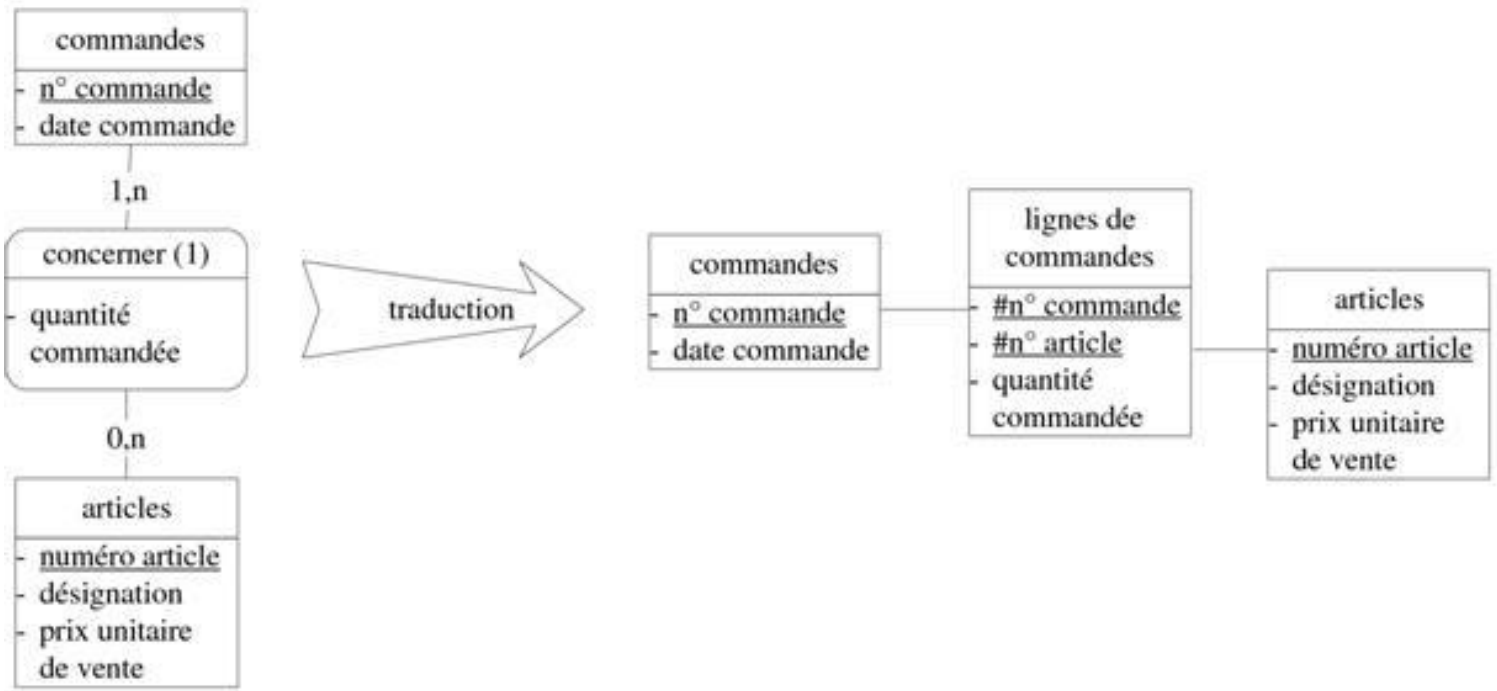


Règle 2 : Association binaire 1 : n (2)

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

Règle 3 : Association binaire $n : m$ (1)

- `lignes de commande(#numero commande, #numero article, quantité commandée)`

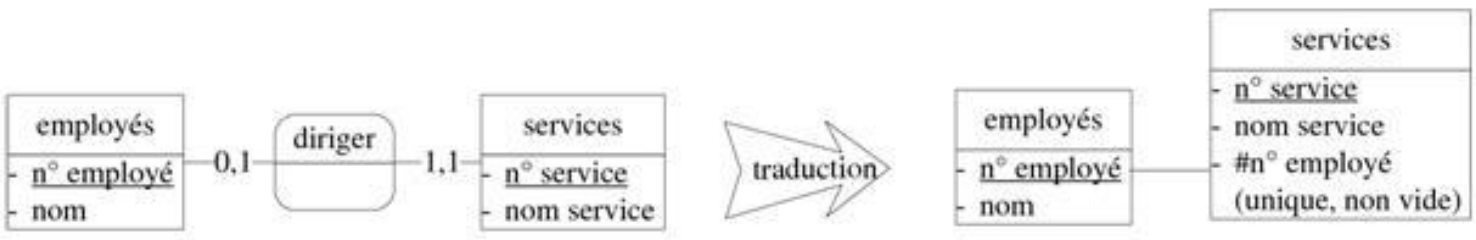


Règle 3 : Association binaire $n : m$ (2)

- Une association binaire de type $n : m$ devient une table :
  - 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

Règle 4 : Association binaire 1 : 1 (1)

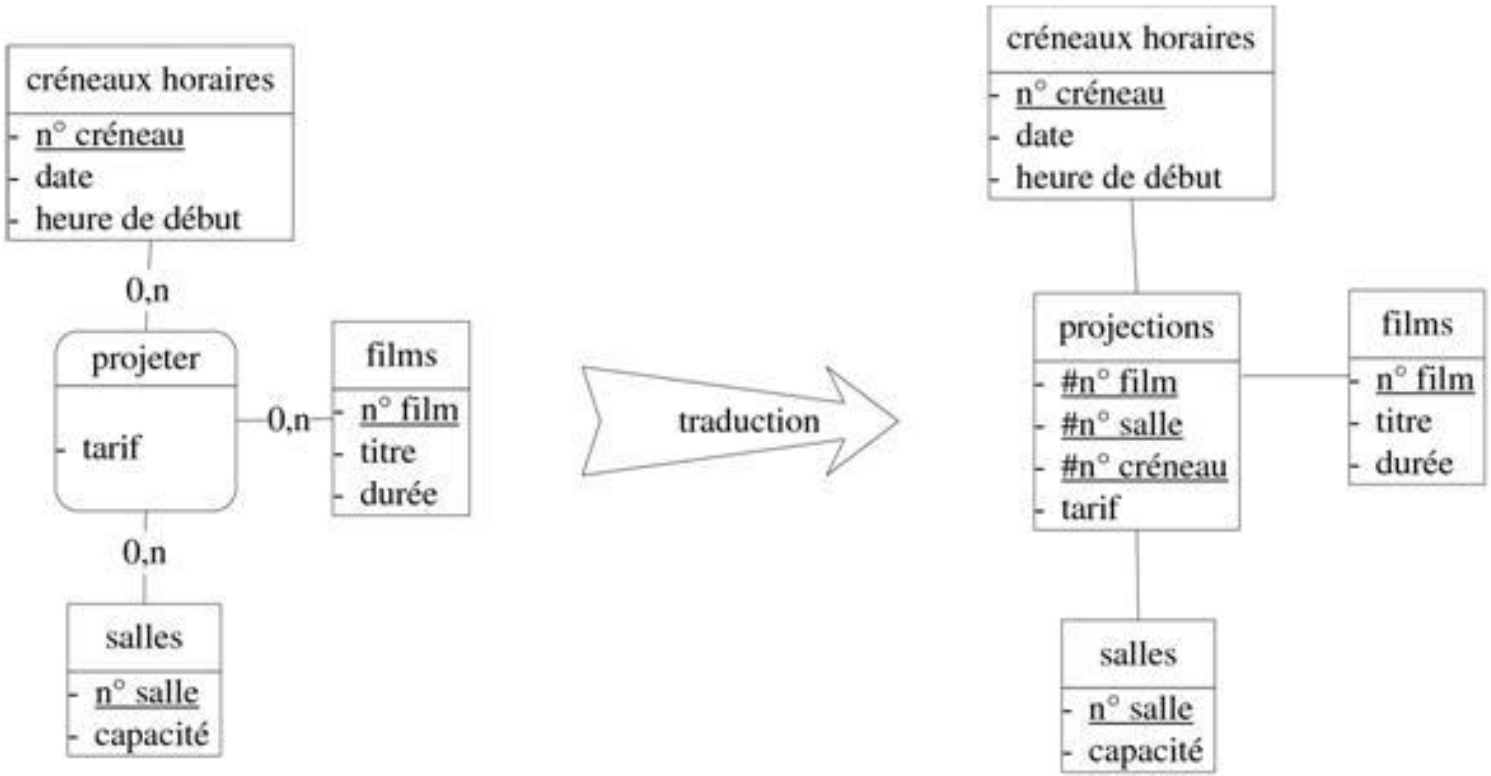
- services(numero service, nom service, #numéro employé (non vide, unique))
- employés(numéro employé, nom)



Règle 4 : Association binaire 1 : 1 (2)

- Une association binaire de type 1 : 1 est :
  - Traduite comme une association binaire de type 1 : n
  - En plus, la clé étrangère se voit imposer une contrainte d'unicité
- 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 placée indifféremment dans l'une des deux tables.

Règle 5 : Association non binaire (1)



Règle 5 : Association non binaire (2)

- Une association non binaire est traduite par une table supplémentaire :
  - 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

Optimisation (1)

- 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 dé-normaliser volontairement le système d'information, avec tous les risques d'incohérence et les problèmes de gestion que cela comporte
- Le conseil le plus précieux, en matière d'optimisation, est de ne jamais optimiser *a priori*, mais toujours *a posteriori* :

$\rightarrow$ C'est-à-dire en réponse à une lenteur que le SGBDR n'est pas capable de résoudre tout seul

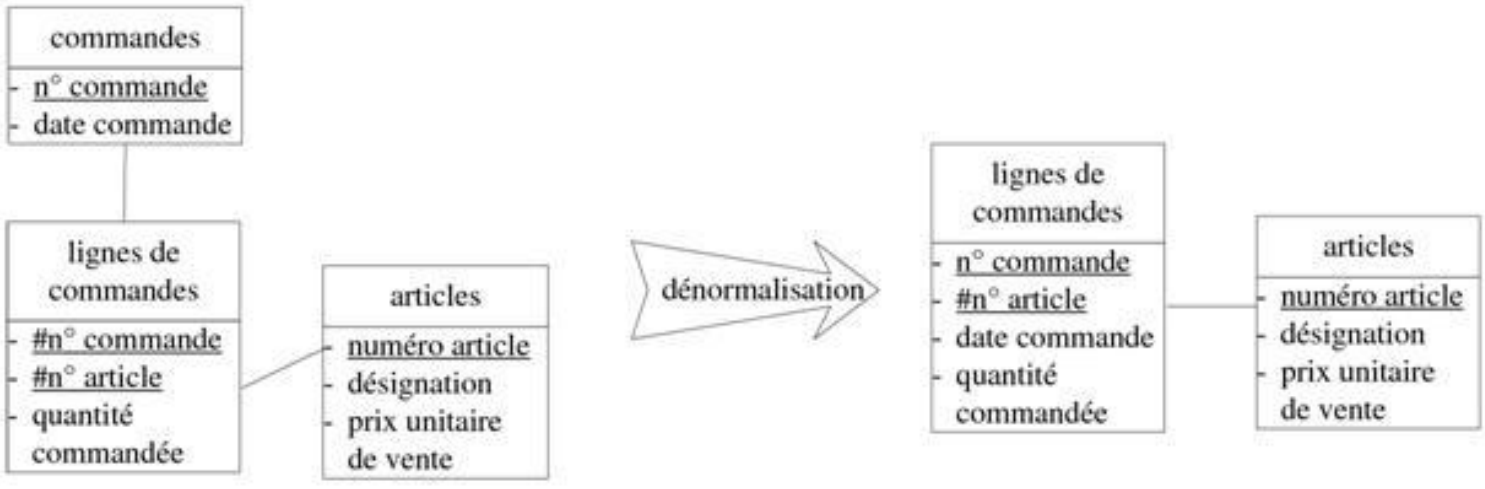
- Il convient de mesurer le gain de toute optimisation manuelle en effectuant des tests (chronométrages avant/après) sur un volume de données significatif et de préférence en exploitation

Optimisation (2)

- Pour les bases de données relationnelles, l'optimisation 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
    - La base de données reste normalisée
  - L'ajout de colonnes calculées ou de certaines redondances :
    - Permet d'éviter des jointures coûteuses
    - La base est dé-normalisée
    - Il faut alors veiller à ce que la cohérence entre les colonnes soit respectée (déclencheurs ou code client)
  - La suppression des contraintes :
    - D'unicité, de clé étrangère, etc.
    - L'intégrité des données doit être assurée par le code client

Optimisation : Exemple

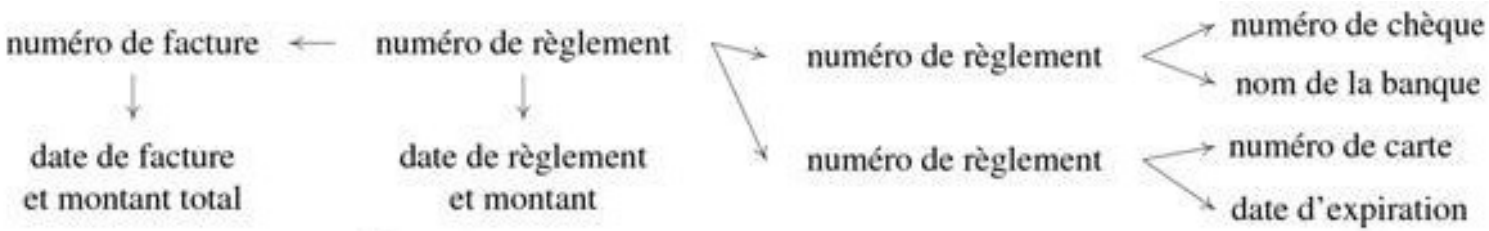
- La table commandes peut être supprimée et la date de commande est alors ajoutée à la table lignes de commandes



- On renonce donc à la troisième forme normale :
  - La date de commande est répétée autant de fois qu'il y a de lignes dans la commande
  - Mais on évite une jointure coûteuse en temps de calcul lors des requêtes SQL

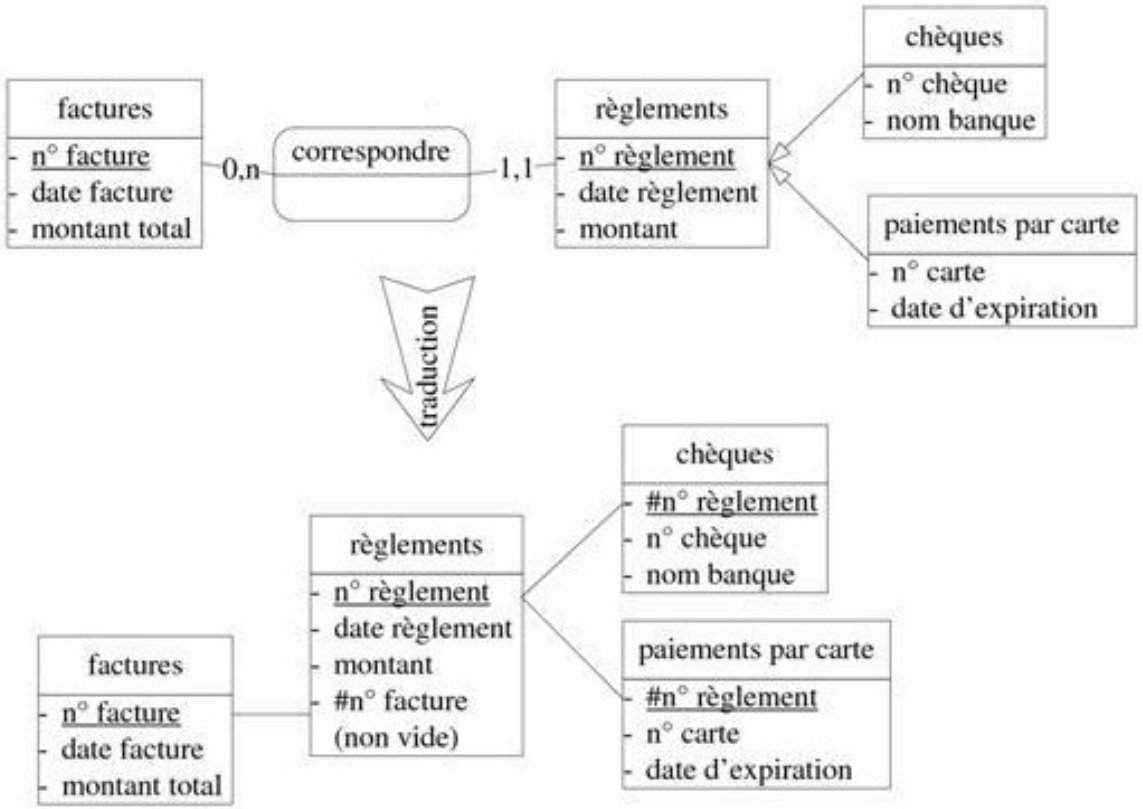
Héritage (1)

- Factorisation des attributs communs à plusieurs entités au sein d'une entité mère



- Exemple :
  - Factures de paiement par chèque ou par carte
  - On souhaite connaître pour chaque règlement la *date* et le *montant*

Héritage (2)



→ Une entité générique règlements et deux entités spécialisées chèques et paiements par carte

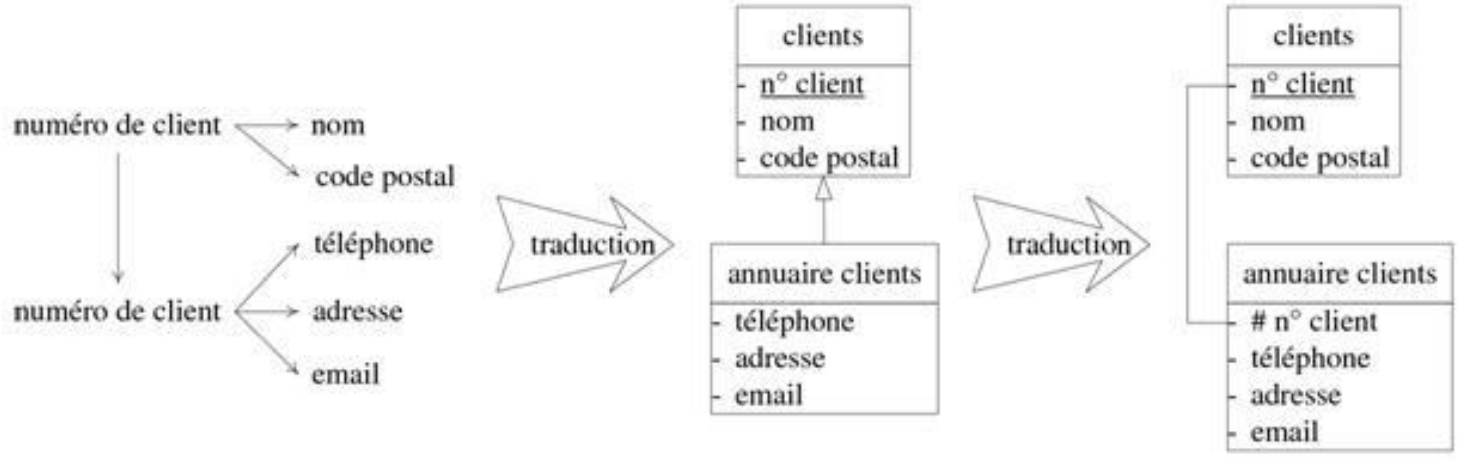
- Lien d'héritage représenté par une flèche creuse (ce lien remplace une association de type 1 : 1)

Héritage (3)

- Les deux sous-entités de l'entité règlements ont des attributs propres mais pas d'identifiant propre
- Il ne faut pas voir d'héritage à chaque fois que l'on peut dire « est un » :
  il faut en plus que l'entité mère ne possède que les attributs communs de ses entités filles
- La traduction des sous-entités au niveau logique relationnel fait intervenir une clé primaire identique à celle de l'entité mère, mais dans les sous-entités la clé primaire est aussi étrangère

Héritage et informations complémentaires

- L'héritage est également utile pour stocker des informations complémentaires



- Exemple :
  - Table clients avec numéro, nom et code postal
  - Ajout de trois colonnes mais il y a déjà des valeurs saisies

  → Une nouvelle table permet de gagner de la place