La conception des bases de données avec UML

Seconde partie

Les gestionnaires de bases de données

Les informations que nous avons collectées précédemment devront être conservées sur un support physique:

- Des fiches cartonnées, à l'origine. (d'où le nom de fichier).
- Des fiches perforées (plus ou moins astucieusement).
- Des supports magnétiques ou optiques aujourd'hui.

Les fichiers permettait alors de conserver des informations mais, le nombre d'informations que l'on stockait était si important que les temps de recherche d'une information précise prenaient étaient prohibitifs

Les gestionnaires de bases de données

La capacité des disques magnétiques ayant augmenté, un certain nombre de logiciels ont été créés afin de pouvoir gérer les données en garantissant l'évolution des informations et leur persistance.

Ces logiciels respectaient l'acronyme ***CRUD***:

- C pour *Create*: permettre l'addition d'une information.
- R pour *Read* : permettre la lecture d'une information.
- U pour *Update* : permettre la mise à jour d'une information.
- D pour *Delete* : permettre la suppression d'une information.

Ce sont les 4 opérations élémentaires de la gestion de données

Les gestionnaires de bases de données

La seconde évolution est liée aux progrès des télécommunications qui ont entrainé la généralisation du télétraitement et donc de l'accès simultané à un centre de calcul par plusieurs utilisateurs.

Sont alors apparus les logiciels de gestion de bases de données.

Qu'est-ce qu'une base de données ?

Les gestionnaires de bases de données

Une base de données est :

- un ensemble structuré de données,
- Interdépendantes,
- Sans redondances,
- Pouvant être utilisées simultanément par plusieurs utilisateurs,
- Garantissant la confidentialité, la cohérence et la sécurité des informations qu'elle contient.

Les gestionnaires de bases de données

Un ensemble *structuré* signifie que chaque information doit être parfaitement identifiée et définie dans ses propriétés:

- Quel est son nom ?
- Cette information est-elle un nombre ?
- Un nombre entier ou décimal ?
- Peut-elle prendre n'importe quelle valeur ?

Il y a tout un tas de conditions à prévoir.

Les gestionnaires de bases de données

Les données *interdépendantes* signifient que la valeur ou l'existence de certaines informations vont dépendre de la valeur d'autres informations qui sont présentes dans la base de données.

Par exemple, dans une base de données médicales, on verrait assez mal renseigner une date d'accouchement alors que la personne est un homme.

Les gestionnaires de bases de données

La *redondance* des données consiste à retrouver plusieurs fois la même information, alors que l'on pourrait s'en dispenser.

Par exemple, si l'on dispose d'une base de données contenant des prix, il sera utile de placer le prix hors taxes et le taux de TVA mais pas le prix TTC, car il serait alors possible que ce prix TTC ne soit pas égal au prix HT affecté de la valeur correcte de la TVA.

Les gestionnaires de bases de données

L'utilisation des données *simultanément* par plusieurs *utilisateurs* va entrainer un certain nombre de problèmes:

- Dans une application de gestion de stock, s'il reste 10 pièces en stock et que, en même temps, deux personnes veulent sortir l'un 8 pièces et l'autre 5 pièces, cela va être impossible.
- Même cas de figure dans un logiciel de gestion des places d'un avion, plusieurs réservations simultanément sur le même vol.

Les gestionnaires de bases de données

La *confidentialité* des informations suppose que n'importe qui ne puisse pas avoir accès à toutes les informations: Cela entraine:

- que certaines informations soient réservées à certains utilisateurs,
- que certaines opérations soient interdites à certains utilisateurs.

C'est ce que l'on appelle donner ou enlever des *droits* à un utilisateur.

Pour faciliter cette gestion, le concept de *groupes d'utilisateurs* a été introduit de façon à regrouper les utilisateurs ayant les mêmes droits.

Les gestionnaires de bases de données

La *cohérence* des informations consiste à s'assurer que les modifications effectuées le sont partout et ce, jusqu'à la fin du processus (d'où la notion de *transaction*).

Exemple d'un transfert d'argent entre deux comptes bancaires:

1. On débite le premier compte.
2. On crédite le second compte.

Que se passe t'il si une coupure de courant se produit entre 1 et 2 ?

Le premier compte est débité, mais le second n'a pas été crédité.

Ou est l'argent ?

Les gestionnaires de bases de données

On définit alors que les opérations complexes appliquées à une base de données devront être ACID:

- A pour Atomicité :

L'ensemble des opérations constitue un tout qui doit être entièrement exécuté, comme un ensemble unique: si un problème se produit (y compris un problème matériel comme une coupure de courant) afin la fin des opérations, les données doivent être replacées dans leur état initial.

Les gestionnaires de bases de données

On définit alors que les opérations complexes appliquées à une base de données devront être ACID:

- A pour Atomicité :
- C pour Cohérence :

La série d'opération doit faire passer l'état de la base de données d'un état cohérent à un autre état cohérent.

Les gestionnaires de bases de données

On définit alors que les opérations complexes appliquées à une base de données devront être ACID:

- A pour Atomicité :
- C pour Cohérence :
- I pour Isolation :

L'exécution de plusieurs transactions qui ont été initiées dans le même laps de temps doit donner le même résultat que si ces transactions avaient été effectuées les unes à la suite des autres.

Les gestionnaires de bases de données

On définit alors que les opérations complexes appliquées à une base de données devront être ACID:

- A pour Atomicité :
- C pour Cohérence :
- I pour Isolation :
- D pour Durabilité :

Le résultat d'une transaction doit être enregistré de façon permanente, sur un support d'information durable et fiable, insensible aux pannes.

Les gestionnaires de bases de données

On définit alors que les opérations complexes appliquées à une base de données devront être ACID:

- A pour Atomicité :
- C pour Cohérence :
- I pour Isolation :
- D pour Durabilité :

Il existe différents modèles de gestionnaires de bases de données …

Les gestionnaires de bases de données

Historiquement, les principaux modèles sont les suivants:

- Le modèle hiérarchique. (Apollo en 1960).
- Le modèle réseau (Bachmann en 1973, supporté par IDS et Pick).
- Le modèle relationnel (Codd en 1970)
- Le modèle objet.

Le modèle le plus utilisé aujourd'hui est le modèle relationnel

Les gestionnaires de bases de données

Les principaux Systèmes de Gestion de Bases de Données Relationnels (SGBDR) utilisés aujourd'hui sont :

- Oracle : pour de très gros projets, très cher.
- MySQL : gratuit, très présent dans les applications Web.
- MariaDb : gratuit, fork de MySQL.
- PostgreSQL : gratuit et open-source

Pour illustrer nos exemples, nous utiliserons MySQL et Workbench qui est un outil d'administration de SGBDR SQL.

Les gestionnaires de bases de données

Les opérations sur les bases de données relationnelles sont effectuées sur le mode client-serveur:

- Un utilisateur envoie une requête au serveur.
- Le serveur analyse la requête et renvoie une réponse.
- L'utilisateur récupère la réponse (qui peut être un message d'erreur si la requête a été mal construite).

Les requêtes sont écrites en suivant les règles d'un langage de requête normalisé (normalement) dont le nom est SQL (Structured Query Language , langage d'interrogation structuré).

La conception d'une base de données relationnelle

Nous allons détailler les diverses notions qui participent à la réalisation d'un modèle conceptuel de données. Il s'agit des notions suivantes:

- Les entités et les Classes Entités
- Les associations et les Classes Association
- Les cardinalités.

La conception d'une base de données relationnelle

Les bases de données relationnelles sont basées sur un ensemble de tables dont les informations sont liées entre-elles.

Il existe plusieurs façons de procéder pour concevoir et représenter une base de données relationnelle:

- Les dépendances fonctionnelles (DF)
- La méthode Merise (ou Entité Associations, E/A).
- La méthode UML (United Modeling Language)

Merise et UML sont très proches l'une de l'autre, UML étant plus général que Merise. Pour les modélisations de bases de données, seules la représentation diffère, celle de Merise étant plus simple à appréhender. Looping permet de passer de l'une des trois représentations à l'autre instantanément.

La conception d'une base de données relationnelle

Notre exemple ne comprends que peu de rubriques primaires. Pour des applications d'une plus grande ampleur, le nombre de rubriques peut atteindre plusieurs centaines.

Il est donc nécessaire de regrouper les rubriques au sein d'entités qui auront la même structure et qui regrouperont des informations ayant un certain rapport entre elles.

Au sein de ces entités, les rubriques prendront le nom de champs et ces rubriques seront identifiées par un identificateur qui sera composé de lettre, de chiffres et du caractère souligné (en commençant par une lettre).

La colonne Formule (inutile) sera remplacée par entité.

La conception d'une base de données relationnelle

Il y a une condition impérative à respecter pour affecter une rubrique à une entité:

La rubrique ne doit avoir qu'une seule valeur.

Par exemple: Si on considère une entité ***Personne*** dans laquelle on souhaite intégrer la rubrique ***Téléphone*** :

Comment traiter une personne ayant plusieurs numéros de téléphone ?

On verra comment procéder plus tard, mais on ne devra pas intégrer la rubrique à l'entité.

La conception d'une base de données relationnelle

Nous allons repartir du dictionnaire de données :

Liste des rubriques

Numéro	Libellé	Propriétés	Identificateur	P/C	Formule
5	Nom ou Raison sociale du client			P	
6	Adresse du client			P	
7	Code Postal et ville du client			P	
8	Numéro de la facture			P	
9	Date de la facture			P	
10	Code de l'article vendu			P	
11	Désignation de l'article vendu			P	
12	Quantité vendue			P	
13	Prix unitaire (HT) de l'article vendu			P	
18	Conditions de règlement			P	

La conception d'une base de données relationnelle

On renomme la colonne Formule en entité:

Liste des rubriques

Numéro	Libellé	Propriétés	Identificateur	P/C	Entité
5	Nom ou Raison sociale du client			P	
6	Adresse du client			P	
7	Code Postal et ville du client			P	
8	Numéro de la facture			P	
9	Date de la facture			P	
10	Code de l'article vendu			P	
11	Désignation de l'article vendu			P	
12	Quantité vendue			P	
13	Prix unitaire (HT) de l'article vendu			P	
18	Conditions de règlement			P	

La conception d'une base de données relationnelle

Quelles sont les entités que l'on peut retrouver dans notre liste:

On voit que plusieurs rubriques se rapportent au ***client***:

Ce sont les rubriques 5,6 et 7.

Quoi d'autre ?

Les rubriques 8 et 9 se rapportent à une ***facture***.

Les rubriques 10,11 et 13 se rapportent à un ***article*** vendu.

On peut compléter le tableau de la façon suivante:

La conception d'une base de données relationnelle

Première mise à jour de la liste des champs :

Liste des rubriques

Numéro	Libellé	Propriétés	Identificateur	P/C	Entité
5	Nom ou Raison sociale du client			P	Client
6	Adresse du client			P	Client
7	Code Postal et ville du client			P	Client
8	Numéro de la facture			P	Facture
9	Date de la facture			P	Facture
10	Code de l'article vendu			P	Article
11	Désignation de l'article vendu			P	Article
12	Quantité vendue			P	
13	Prix unitaire (HT) de l'article vendu			P	Article
18	Conditions de règlement			P	

La conception d'une base de données relationnelle

Dans notre exemple de facture, la mairie de Allongville est un client de la société « Ecole des Super Cracks ».

C'est une entité.

Mais, si on considère une autre facture, relative à la mairie de Retourville, par exemple, cette autre mairie sera également une entité.

Ces deux entités auront les mêmes champs qui contiendront des valeurs différentes, mais elles auront les mêmes champs.

On dira que ces deux entités font partie de la même ***classe***, que l'on pourra appeler la classe ***Clients***.

La conception d'une base de données relationnelle

Dans une classe entité, il va y avoir plusieurs entités (La classe Clients va contenir plusieurs références à des clients différents). On parle d'instance de classe.

Pour permettre une bonne gestion de la classe, il est impératif de disposer d'un moyen pour retrouver une instance particulière grâce à un (ou plusieurs) champs de la classe.

Si je considère une classe ***Elèves*** contenant tous les élèves de la classe, comment faire pour donner une note à quelqu'un.

Quel est le (ou les) champ(s) que je devrais utiliser ?

Cet ensemble de champs va s'appeler l'identifiant (ou la clé primaire) de la classe. L'identifiant est souligné (Merise) ou en **gras** (UML).

La conception d'une base de données relationnelle

On en tire une seconde règle:

Toute classe entité doit avoir un identifiant.

La conception d'une base de données relationnelle

En notation Merise:

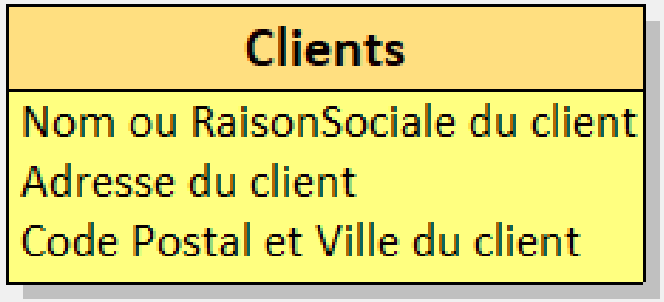
Clients
Nom ou RaisonSociale du client
Adresse du client
Code Postal et Ville du client

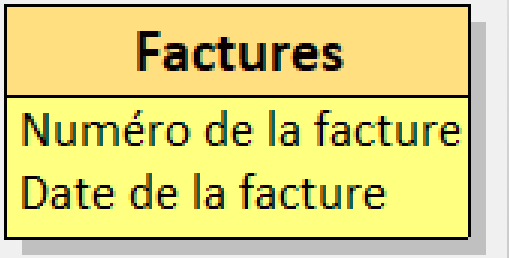
Factures
Numéro de la facture
Date de la facture

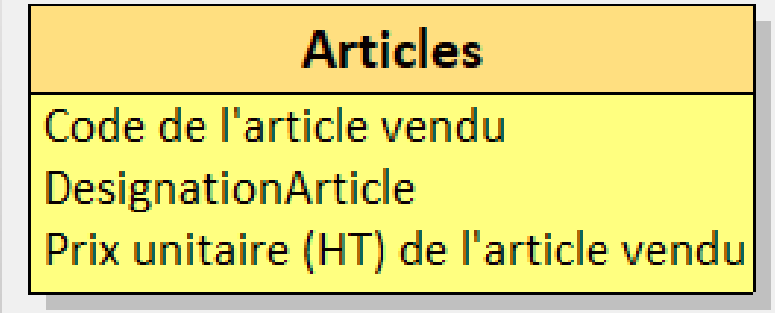
Articles
Code de l'article vendu
DesignationArticle
Prix unitaire (HT) de l'article vendu

La conception d'une base de données relationnelle

En notation Merise: N'y a t-il pas un problème ?

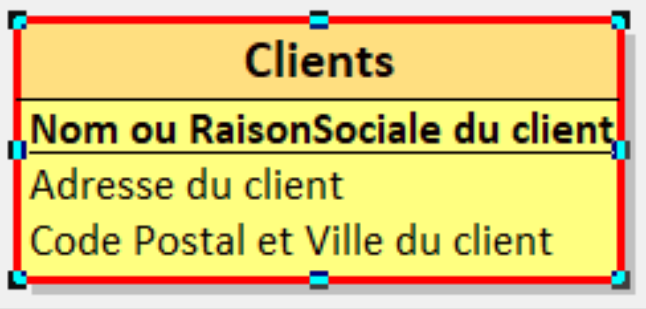


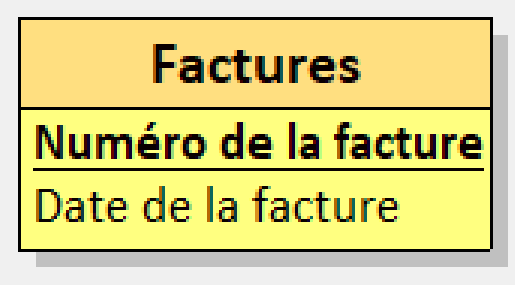


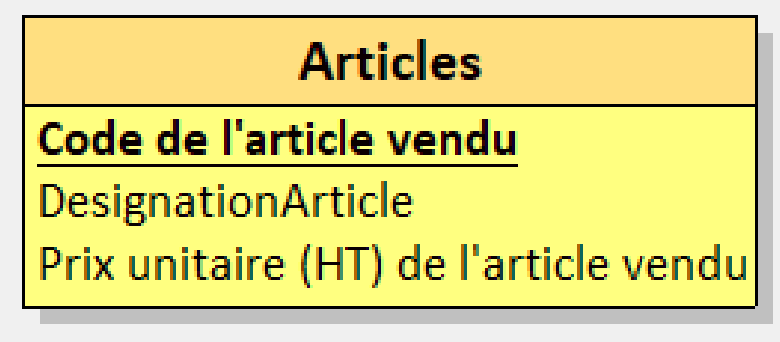


La conception d'une base de données relationnelle

En notation Merise: On définit les clés

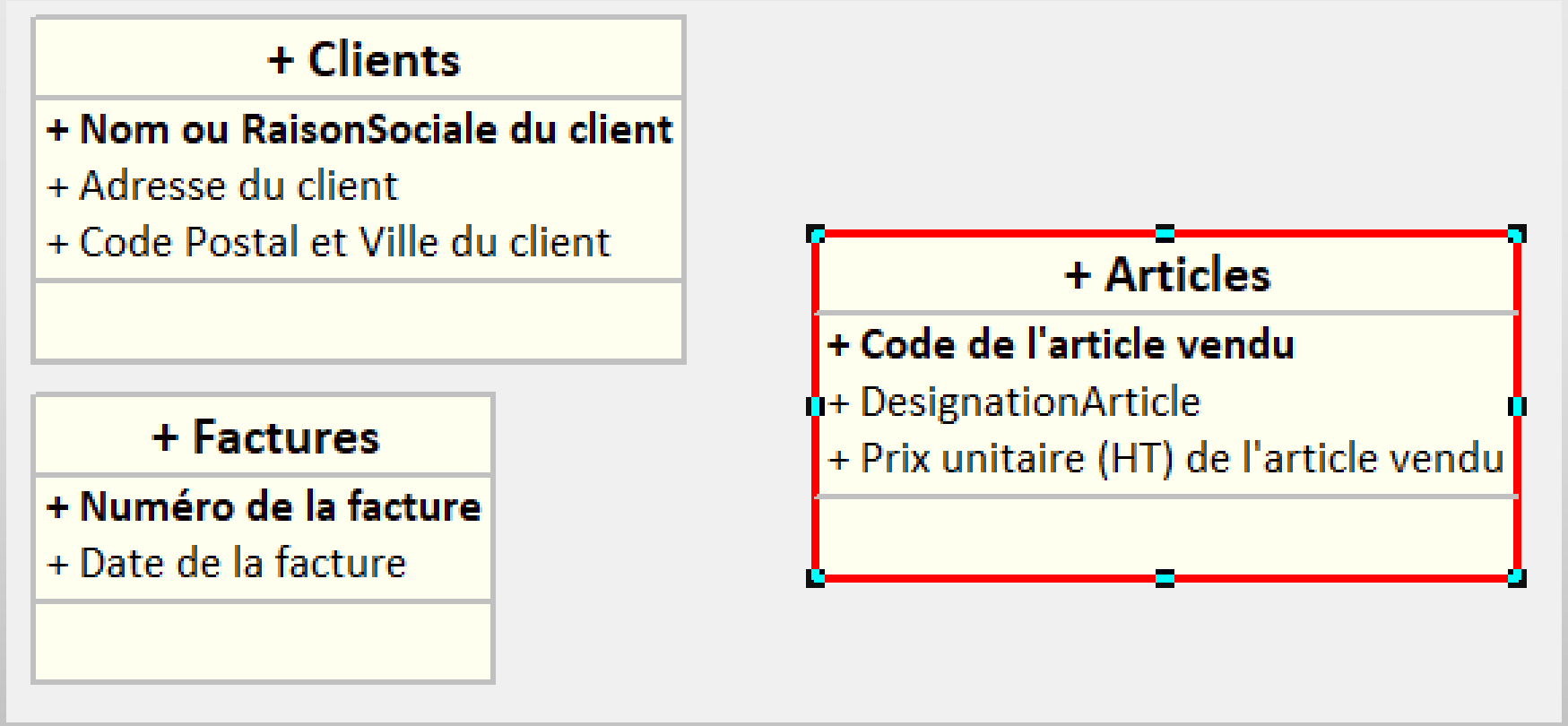






La conception d'une base de données relationnelle

En notation UML :



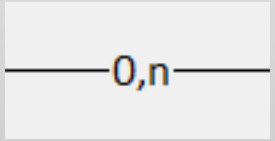
La conception d'une base de données relationnelle

La notation Merise comprend les éléments suivants:

- Les entités:

- Les associations :

- Les liens avec leurs cardinalités :

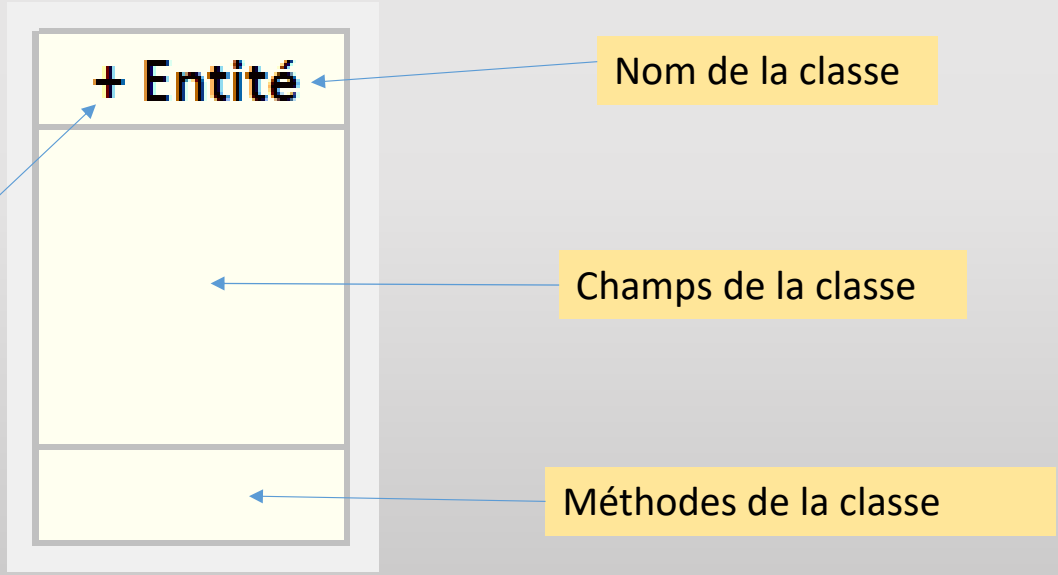


Les cardinalités (ou multiplicités) précisent le nombre minimum et le nombre maximum d'associations possibles entre deux entités.

La conception d'une base de données relationnelle

La notation UML est de la forme suivante :

- Les classes:



La conception d'une base de données relationnelle

La notation UML est de la forme suivante :

- Les multiplicités:

La syntaxe demande de préciser le minimum et le maximum possible de la façon suivante : *minimum* .. *Maximum*

Exemple : Entre 2 et 5 (bornes comprises) s'écrira 2..5

Il y a des raccourcis:

* À la place de maximum permet de ne pas préciser de maximum.

n..n s'écrira n

0..* s'écrira *

La conception d'une base de données relationnelle

Dans notre exemple de facture, il reste deux champs qui ne sont reliés à aucune entité, les champs 12 et 18.

Considérons d'abord le champ 18 « Conditions de règlement » :

Une question se pose:

Tous les clients sont-ils soumis aux mêmes condition de règlement ?

Cela va dépendre des circonstances, de la politique de la société, de ce que l'on appelle les ***règles de gestion***.

Ces règles sont en général précisées par les responsables de l'organisation, et, en cas de doute, il faudra les faire apparaitre dans un document contractuel.

La conception d'une base de données relationnelle

Dans notre exemple de facture, il reste deux champs qui ne sont reliés à aucune entité, les champs 12 et 18.

Considérons d'abord le champ 18 « Conditions de règlement » :

Si tous les clients sont soumis aux mêmes conditions : La rubrique correspondante est un paramètre, et ne doit donc pas faire partie de la liste des champs; le problème est réglé.

Sinon, et ce sera notre choix, chaque client aura ses propres conditions de règlement et donc le champ fera partie de la classe ***Clients***, et ce sera possible parce qu'un client n'aura qu'un seul mode de règlement.

La conception d'une base de données relationnelle

Dans notre exemple de facture, il reste deux champs qui ne sont reliés à aucune entité, les champs 12 et 18.

Il reste le champ 12 « quantité vendue » :

Peut-on mettre ce champ dans la classe ***Clients*** ?

Non, bien sur, mais pourquoi ?

On le voit bien sur la facture : les quantités ne sont pas les mêmes en fonction des articles vendus.

La règle ne sera pas respectée.

La conception d'une base de données relationnelle

Dans notre exemple de facture, il reste deux champs qui ne sont reliés à aucune entité, les champs 12 et 18.

Il reste le champ 12 « quantité vendue » :

Peut-on mettre ce champ dans la classe ***Articles*** ?

Non, bien sur, mais pourquoi ?

Dans une autre facture, un même article (par exemple le code 457) pourra être vendu avec une autre quantité que 12.

La règle ne sera pas respectée.

Comment faire dans ce cas ?

La conception d'une base de données relationnelle

Dans notre exemple de facture, il reste deux champs qui ne sont reliés à aucune entité, les champs 12 et 18.

Il reste le champ 12 « quantité vendue » :

On peur remarquer que, pour un produit donné **ET** une facture donnée, la quantité vendue ne prends qu'une seule valeur.

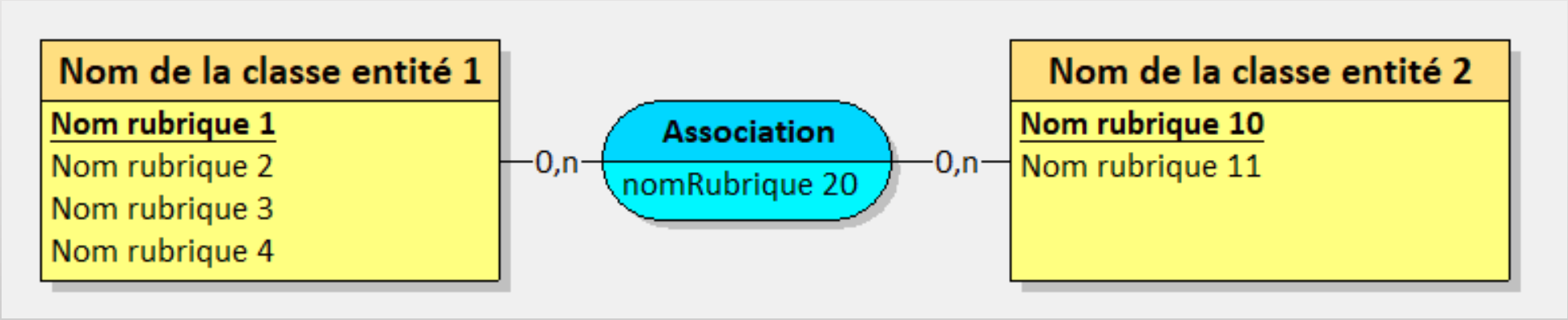
Dans cette situation, la règle est respectée.

Il suffira de créer un élément que l'on placera entre les entités **Produits** et **Factures** dans lequel on placera la rubrique. Cet élément portera le nom de ***Classe Association*** car il associe deux ***Classes Entités***.

La conception d’une base de données relationnelle

La représentation des classes association diffère suivant les notations Merise et UML:

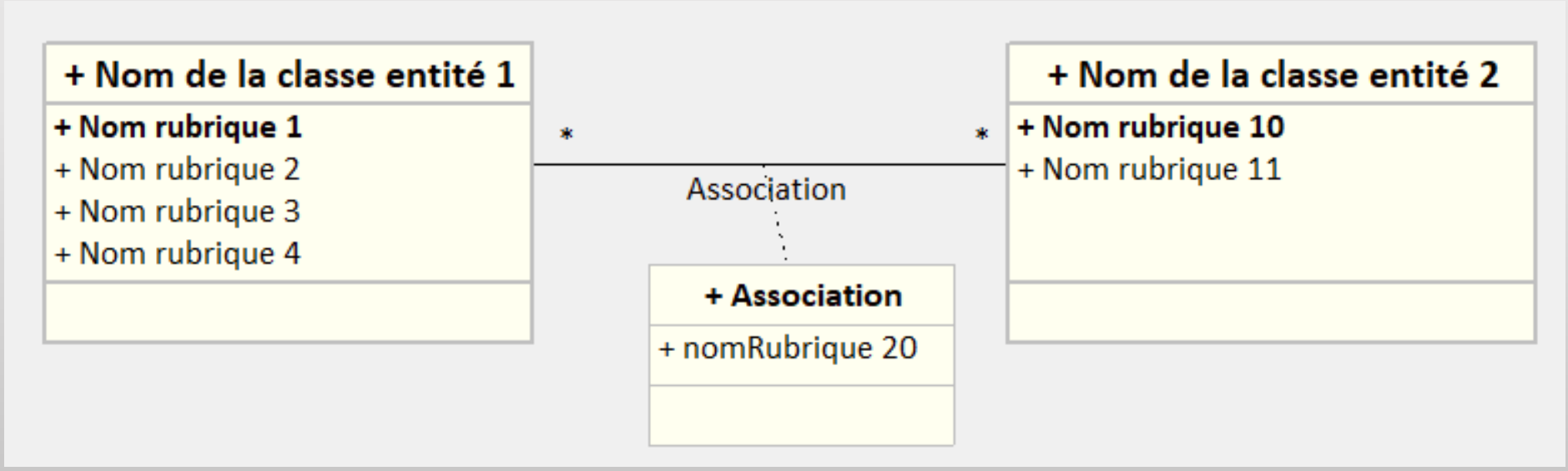
Sous Merise:



La conception d'une base de données relationnelle

La représentation des classes association diffère suivant les notations Merise et UML:

Sous UML :



La conception d'une base de données relationnelle

Dans notre exemple, il faudra donner un nom à la Classe Association que l'on va créer pour associer les factures aux produits.

D'une manière générale, cette association peut (et devrait) être décrite par un verbe.

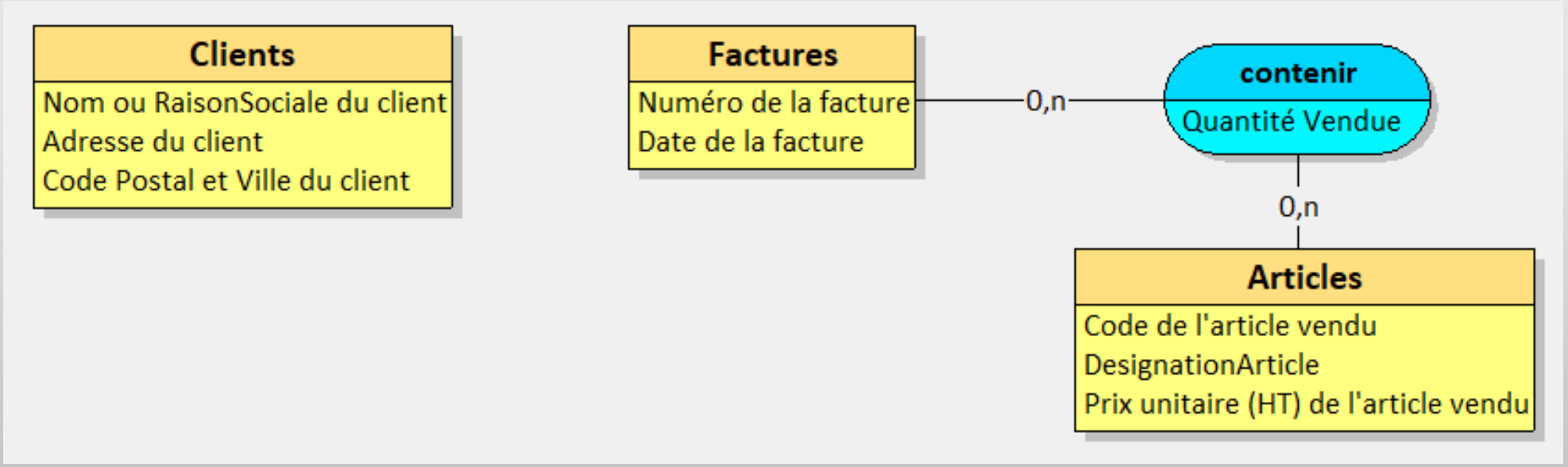
En effet, on pourra dire:

- Une facture contient des produits. (voie active)
- Un produit est contenu dans une facture. (voie passive)

Notre modèle conceptuel deviendra alors:

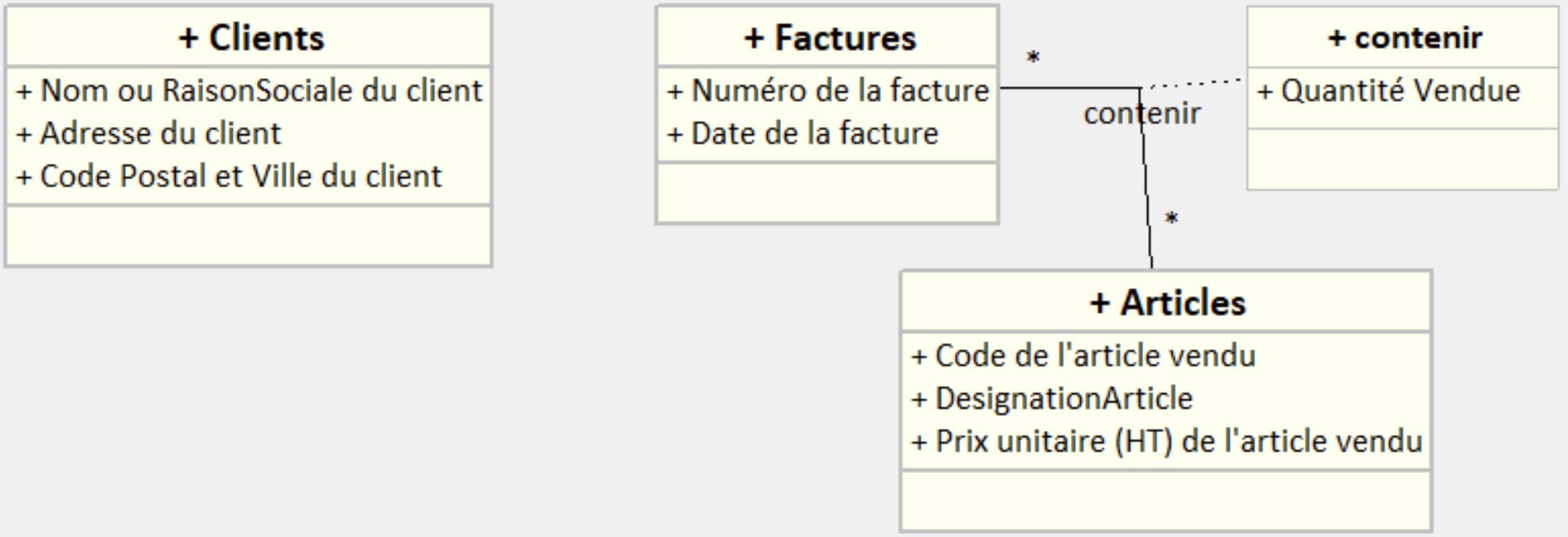
La conception d'une base de données relationnelle

En Merise :



La conception d'une base de données relationnelle

En UML :



La conception d'une base de données relationnelle

Pour résumer:

- Toutes les rubriques qui ne sont pas des paramètres ou des rubriques calculées doivent appartenir à une classe (d'entités ou d'association).
- Une rubrique affectée à une entité ne doit avoir qu'une seule valeur.
- Toute classe entité doit avoir un identifiant.

La conception d'une base de données relationnelle

Normalement, toutes les informations présentes dans le M.C.D. doivent être associées les unes aux autres.

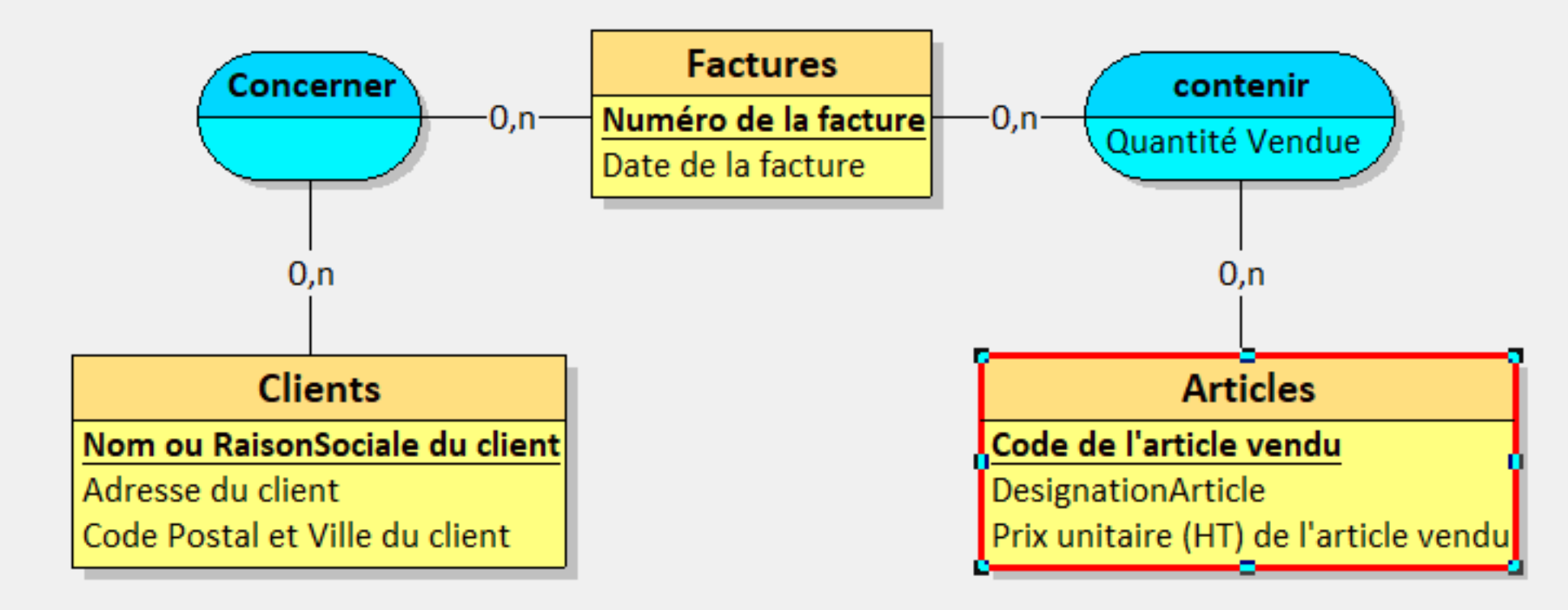
On voit bien, dans l'état actuel de notre exemple, que ce n'est pas le cas pour les informations de la classe Clients.

Or, les clients sont concernés par des factures;

Il est donc logique d'insérer une classe d'association, sans rubrique, entre les classes Clients et Factures.

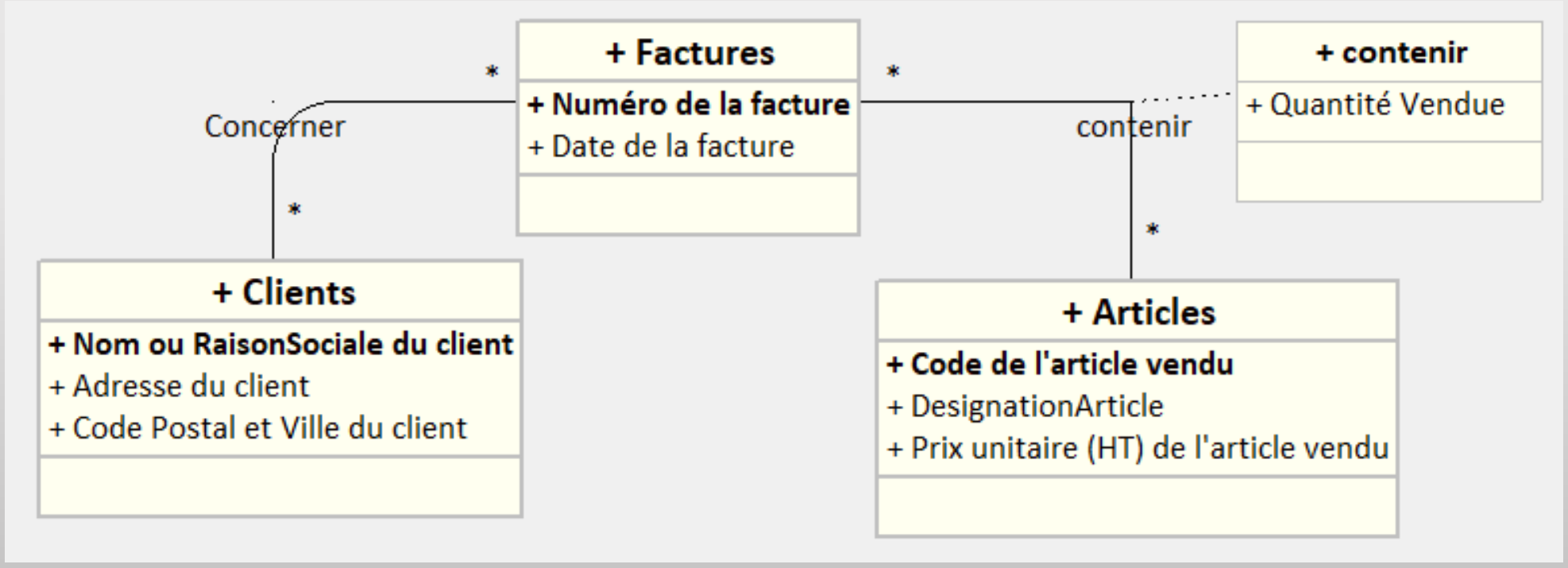
La conception d'une base de données relationnelle

Le M.C.D., en Merise, devient :



La conception d'une base de données relationnelle

Le M.C.D., en U.M.L., devient :



La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)

C'est une notion fondamentale pour la finalisation du modèle:

La cardinalité d'un lien entre deux entités est constitué de deux informations, qui sont, dans l'ordre:

1. Le nombre *minimum* de fois qu'une instance d'une entité est associée à d'autres instances de l'autre entité.
2. Le nombre *maximum* de fois qu'une instance d'une entité est associée à d'autres instances de l'autre entité.

La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)

Dans la notation Merise, les possibilités sont limités aux cas suivants:

1. Minimum 0, maximum égal à 1 : noté 0,1
2. Minimum 0, maximum supérieur à 1 : noté 0,n
3. Minimum supérieur à 0,Maximum égal à 1 : 1,1
4. Minimum supérieur à 0,Maximum supérieur à 1 : 1,n

Dans les exemples précédents, nous n'avons rien précisé à ce sujet, Looping à choisi la plus grande plage : 0,n

La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)

Dans la notation U.M.L., les options sont plus précises, on précise, par des nombres entiers, les valeurs du minimum et du maximum:

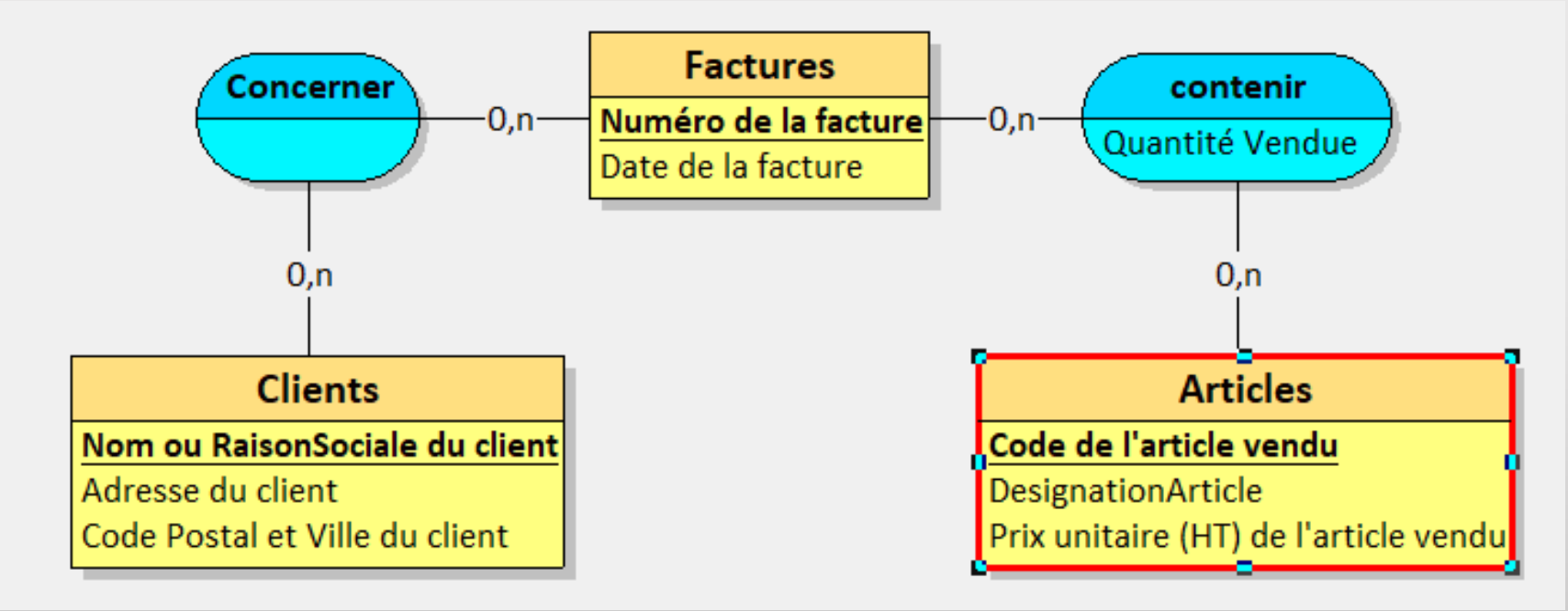
Minimum égal à x, maximum égal à y : noté x..y

Il y a des cas particuliers:

Une étoile (*) à la place d'un nombre signifie plusieurs, sans préciser: par exemple 1..* (mais 0..* pourra se noter simplement *).

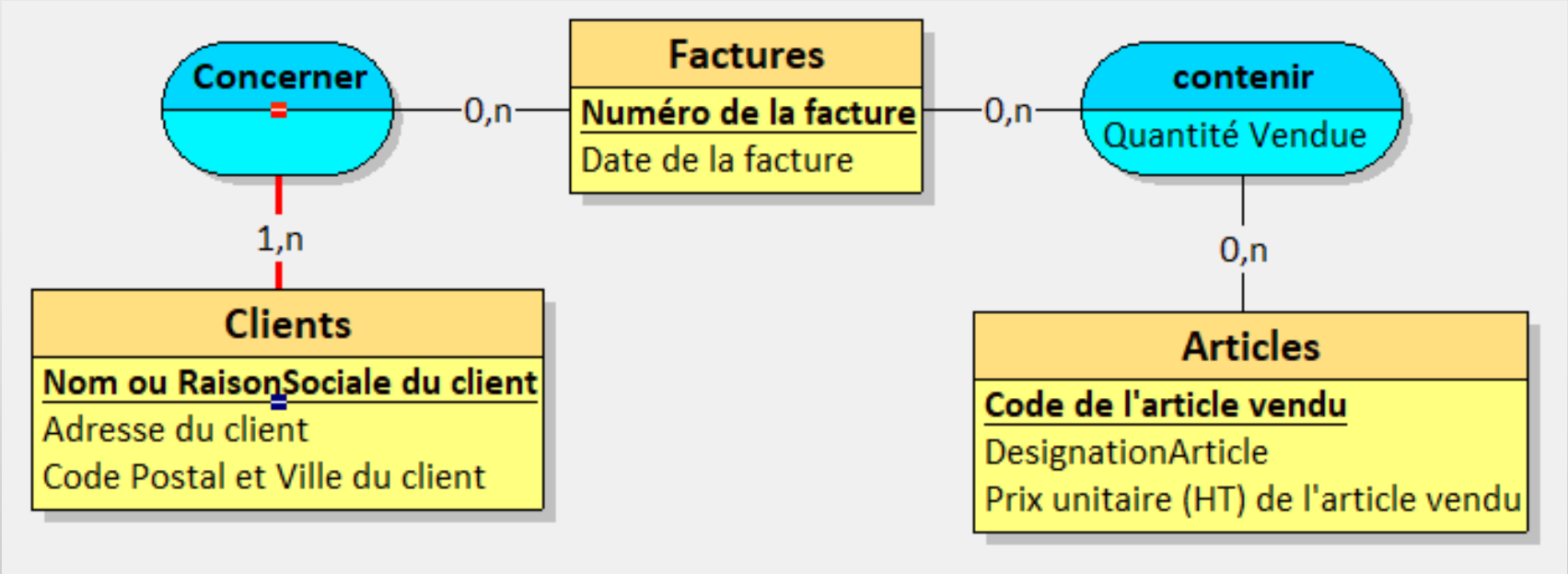
n..n pourra se noter simplement n

La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)



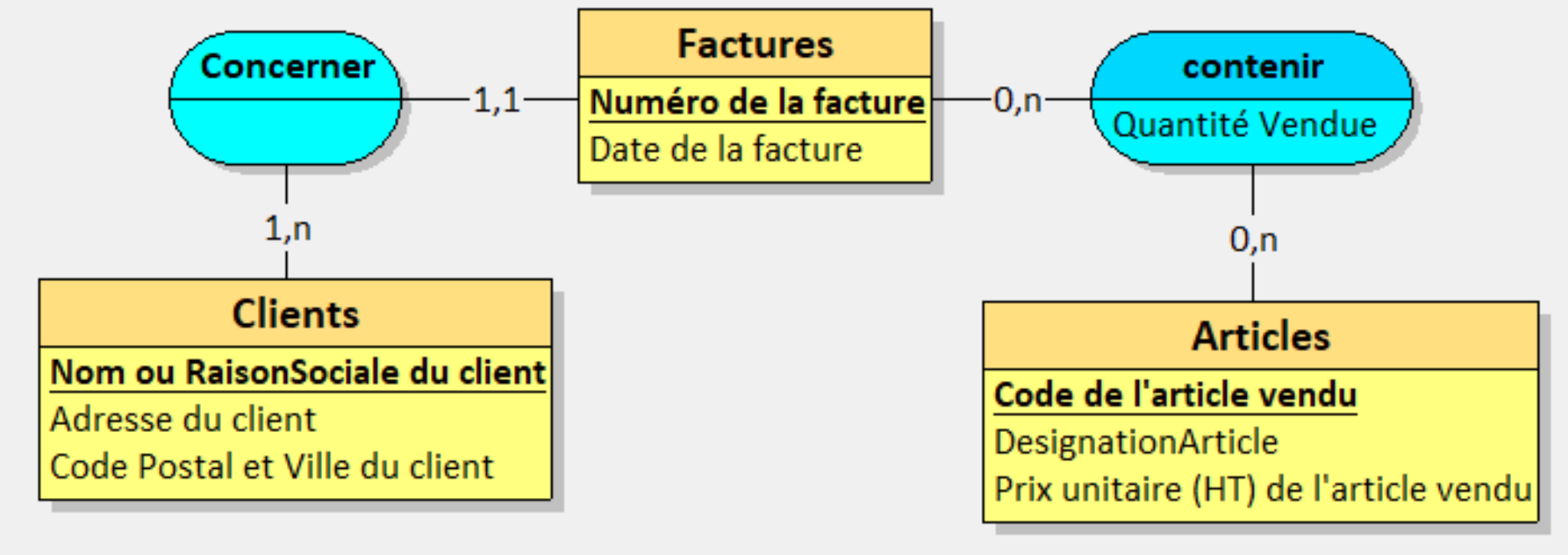
Quelle est la cardinalité entre clients et concerner ?

La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)



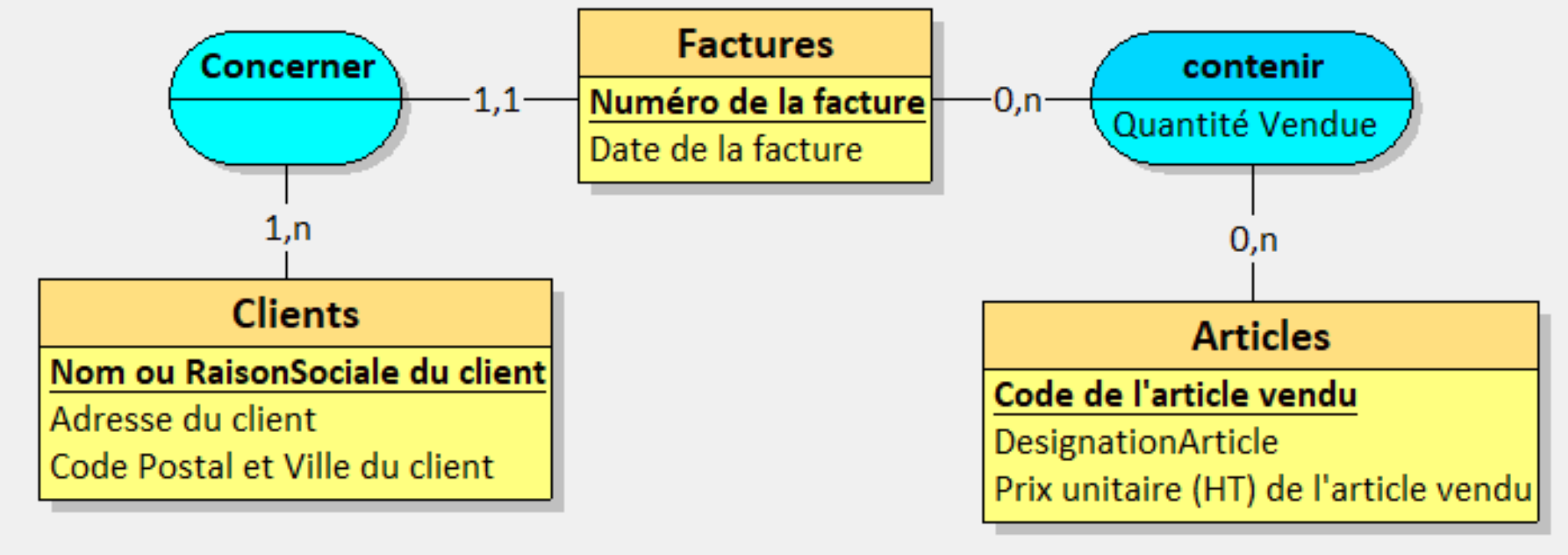
Quelle est la cardinalité entre concerner et factures ?

La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)



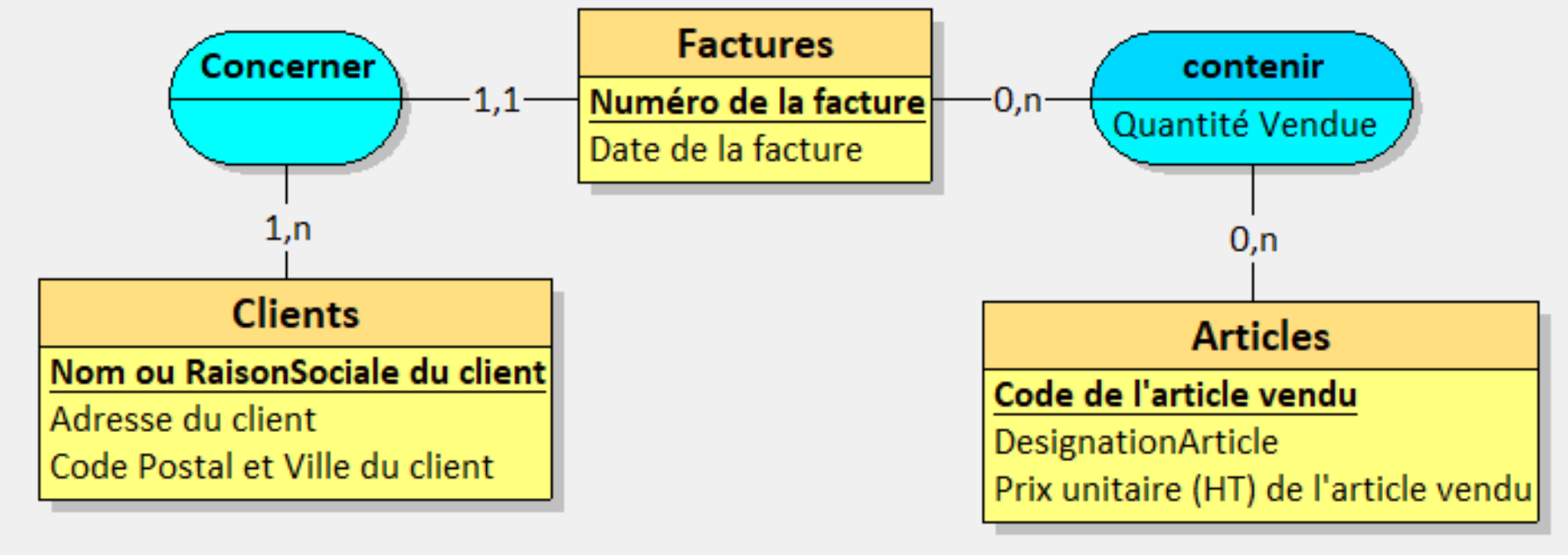
Quelle est la cardinalité entre factures et contenir ?

La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)



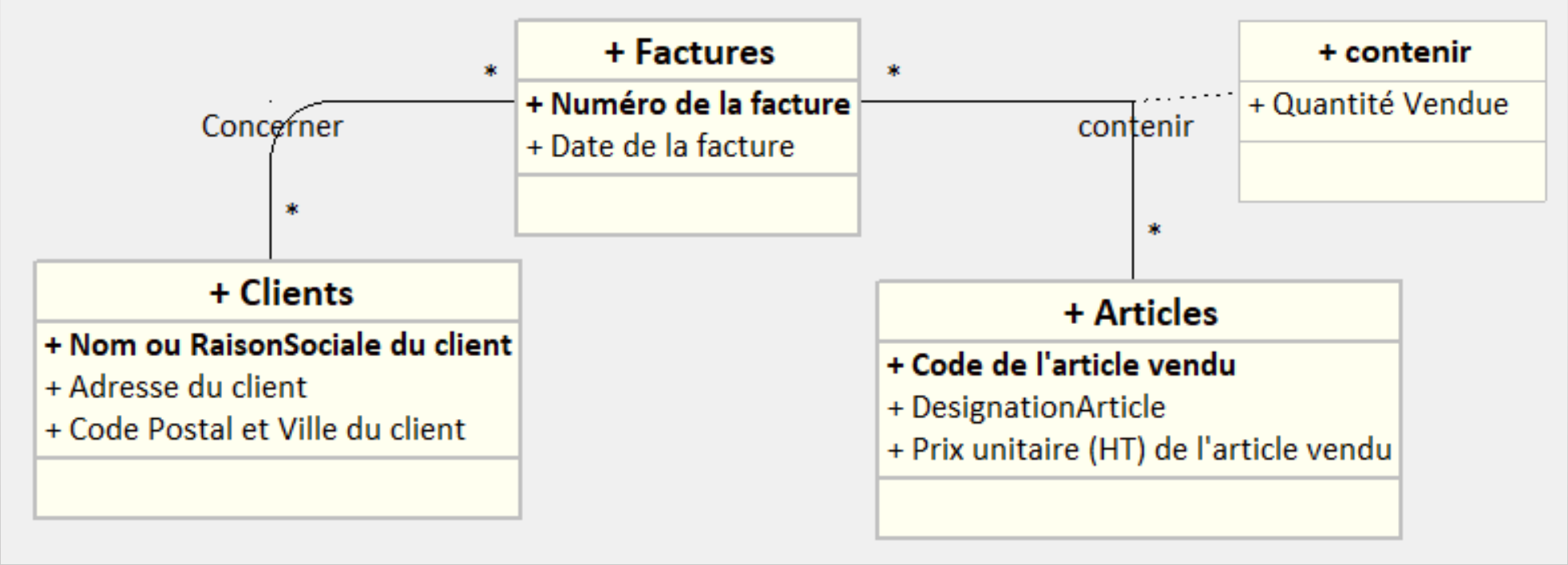
Quelle est la cardinalité entre contenir et articles ?

La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)



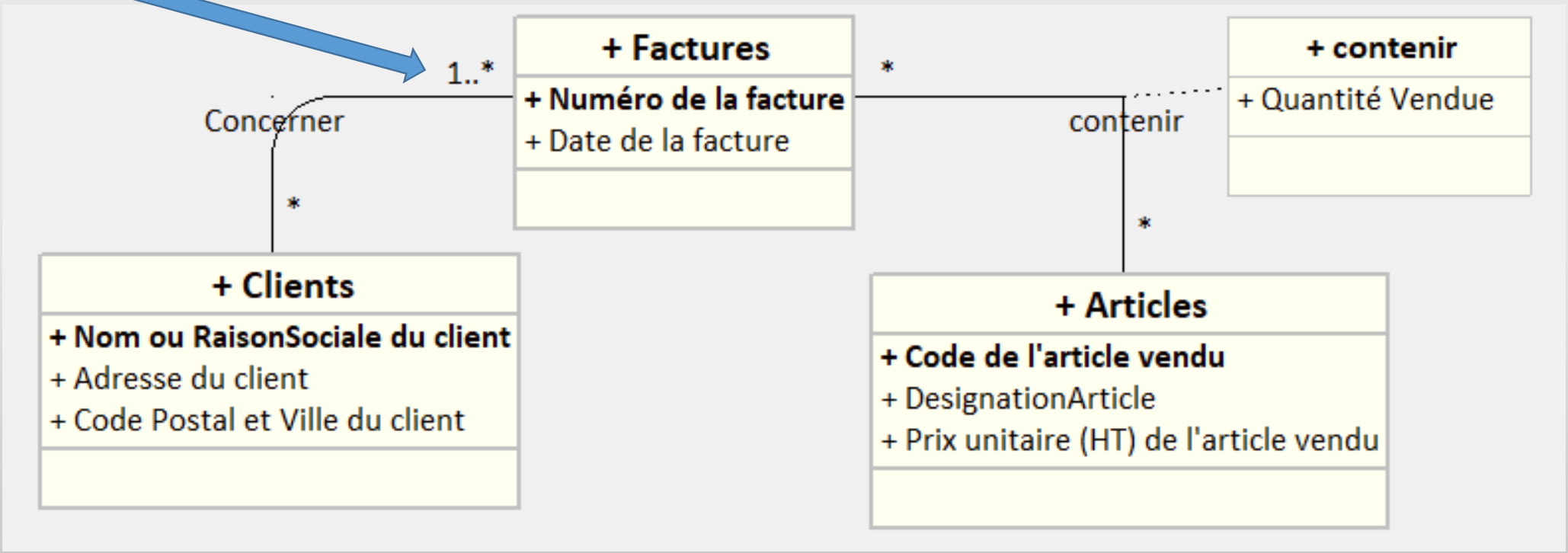
Le M.C.D. est terminé, sous Merise, On refait en U.M.L.

La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)



Quelle est la cardinalité de clients vers factures ?

La conception d'une base de données relationnelle
Les cardinalités (ou multiplicités)



Quelle est la cardinalité de factures vers clients ?