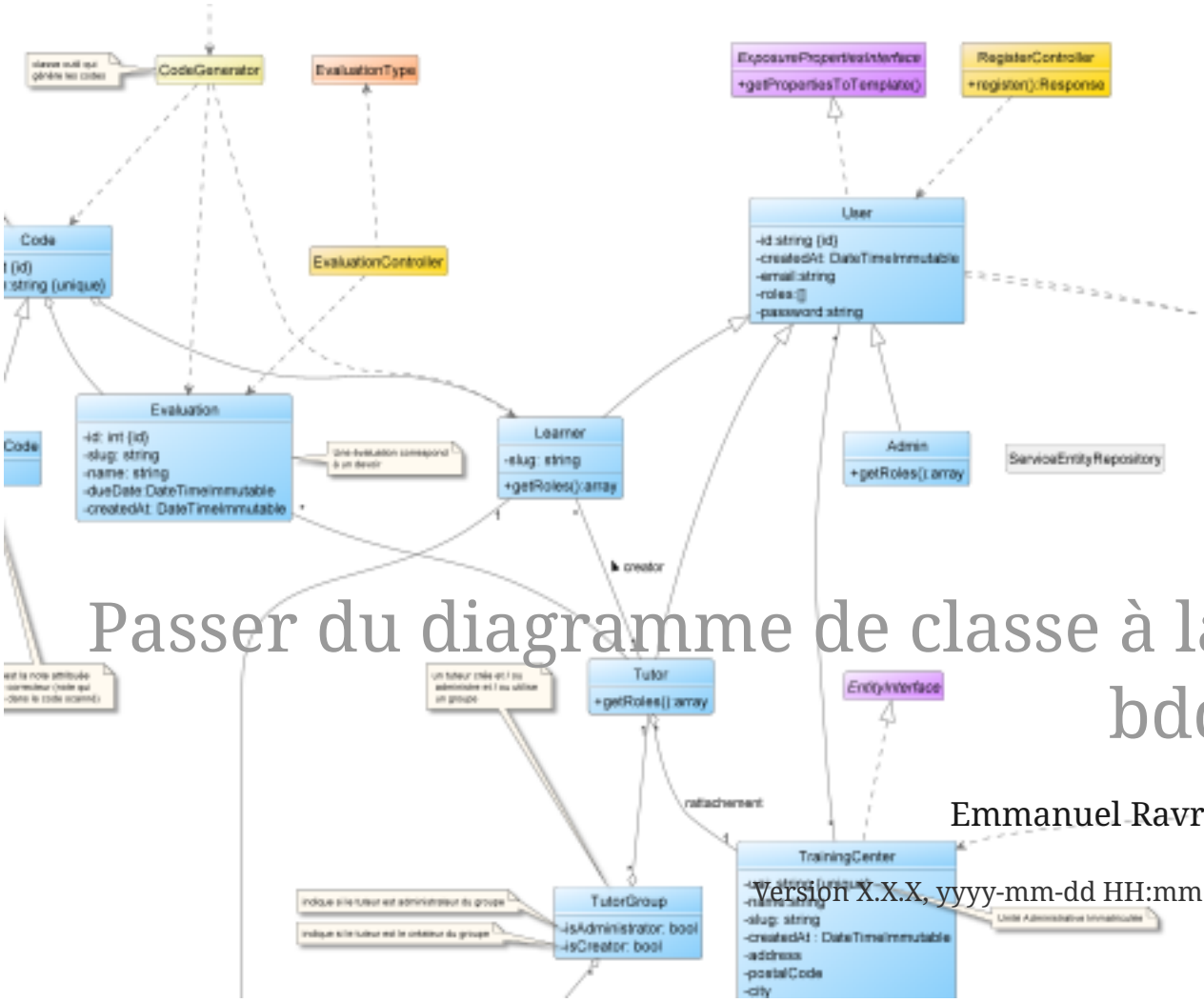


Passer du diagramme de classe à la bdd

Emmanuel Ravrat

Version X.X.X, yyyy-mm-dd HH:mm:ss

Table des matières

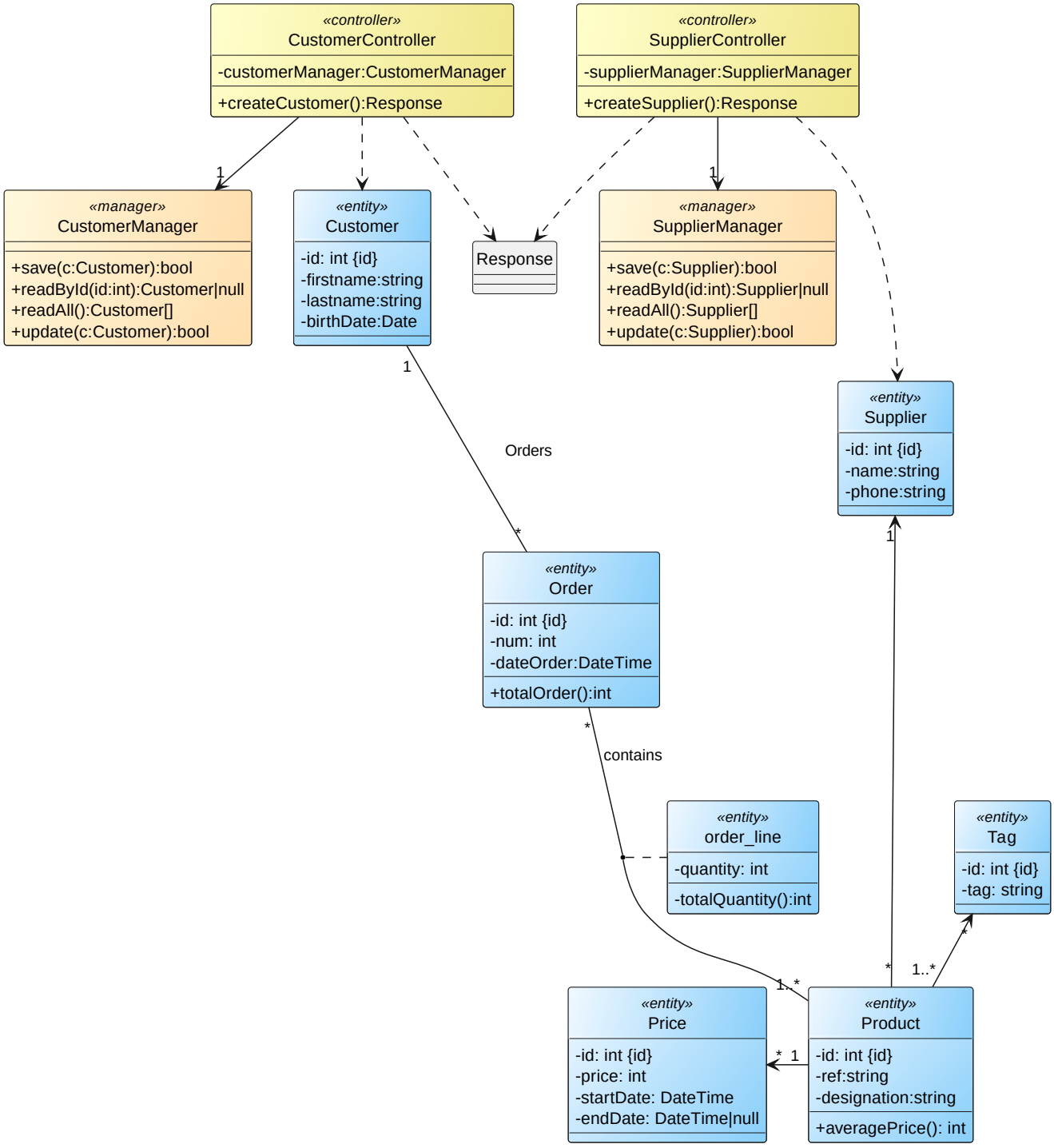
Point précédent : Les interfaces

1. Lien entre diagramme de classe et bdd

Le diagramme de classe est utilisé pour modéliser une application. Certaines classes donnent des instances qui doivent être enregistrée en bdd. Cela s'appelle la **persistance des données**. En programmation, ces classes sont appelées des **entités**.

Dans un diagramme de classe, ces classes sont marquées avec le stéréotype `entity`. Grâce à cela, le développeur qui lit le diagramme sait qu'il doit gérer la persistance des instances de cette classe.

Prenons en compte l'extrait de diagramme ci-dessous qui modélise une application e-commerce (nous allons le construire ensemble) :



Lorsqu'une classe est marquée avec le stéréotype `entity`, il faut déterminer un **identifiant**.

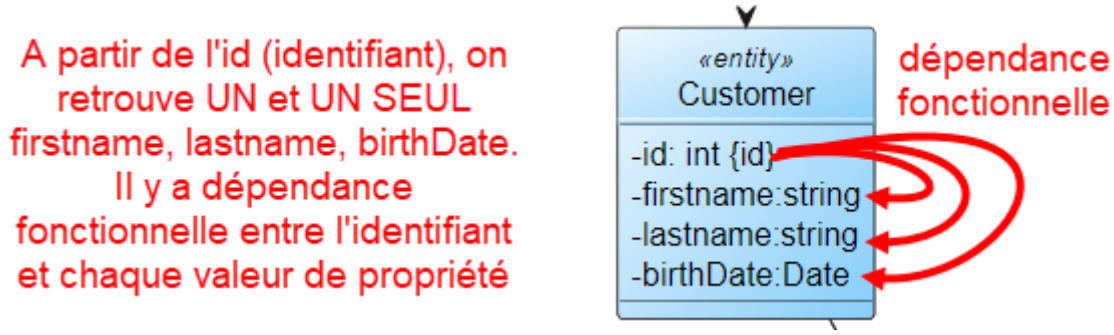
Généralement, comme les applications utilisent un ORM, l'identifiant est un entier **autoincrémenté** [1].

Par exemple, pour l'entité `Customer`, l'**identifiant naturel** pourrait être le "numéro de client". Mais, comme on utilise généralement un ORM, il faut préférer le nom `id` pour "identifiant".

Si les ORM utilisent un identifiant numérique autoincrémenté, c'est parce que cela permet de s'assurer que la valeur qui désigne l'élément inséré est **unique**. En réalité, c'est le système de gestion de base de données qui va se charger de générer cette valeur.

Ainsi, même si deux clients ont le même nom, il sera possible de les différencier grâce à leur identifiant. Dans une table, on appelle cet identifiant une **clé primaire**

La clé primaire (ou identifiant) permet de retrouver de **façon unique** la valeur de chaque propriété :



Comme l'indique le schéma, il y a **dépendance fonctionnelle** entre l'identifiant et chaque valeur associée à cet identifiant.

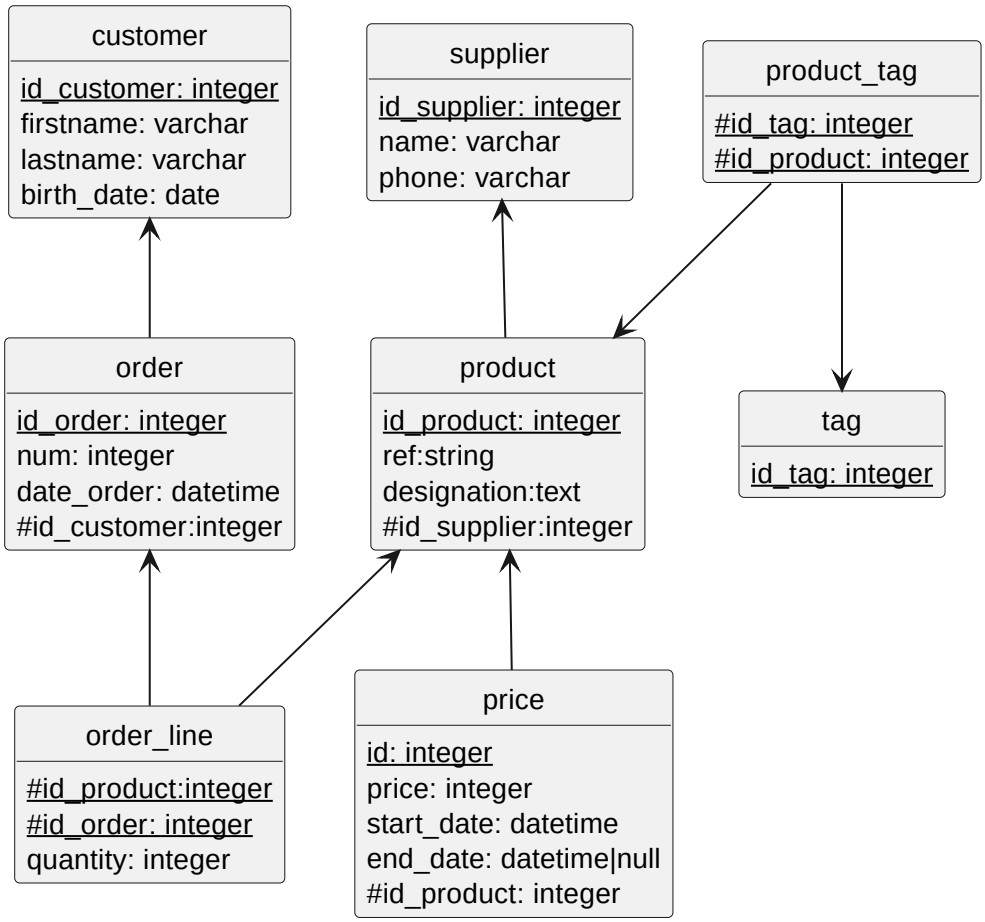
[1] dont la valeur est égale à la valeur précédente + 1

2. Passer du diagramme UML au schéma relationnel

2.1. Qu'est-ce qu'un schéma relationnel ?

Le **schéma relationnel** va permettre de mettre en évidence les tables de la base de données à créer et les **relations** qu'il y a entre elles.

Voici à quoi ressemble le schéma relationnel du diagramme de classe précédent :



Sans se soucier pour l'instant de la méthode qui nous a permis d'obtenir ce schéma relationnel, nous pouvons déduire de sa lecture les informations suivantes :

- il y a 8 tables (ce sont les rectangles)
- pour chaque table, l'identifiant est souligné
- les références entre tables sont représentées par des flèches et l'origine de celles-ci est représentée par le nom d'une propriété précédé par le caractère `#`
- il n'y a pas de cardinalité
- les noms des classes et des propriétés adoptent la notation snake_case (`startDate` devient `start_date`.
- les types de chaque propriété est un type géré par la base de données et pas par l'application (ex

: un type `string` devient un type `varchar`, un long texte avec un type `string` devient un type `text`)

Il est important de comprendre que le schéma relationnel représente la future base de données. Il n'a plus rien à voir avec le diagramme de classe.

Un schéma relationnel n'est d'aucune utilité pour développer une application (bon, j'exagère, il indique tout de même les entités qui doivent être prévues sous forme de classe).

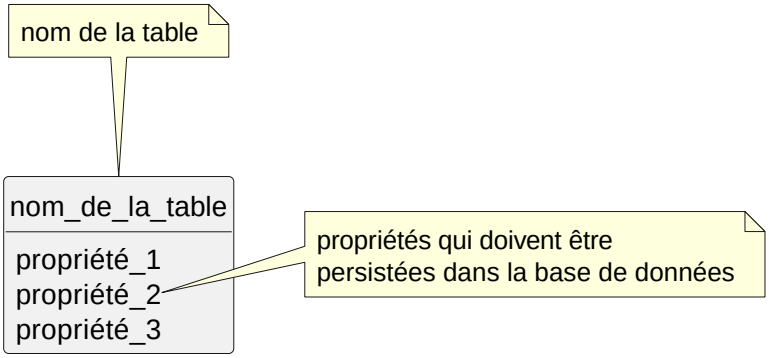
2.2. Règle de passage du diagramme de classe au schéma relationnel

En appliquant quelques règles, il est facile de passer du diagramme de classes au schéma relationnel.

2.2.1. Identifier les futures tables et leurs colonnes

Les classes qui sont marquées par le stéréotype `entity` deviennent des tables.

Dans le schéma relationnel, une table [1] est représentée par un rectangle en deux parties.

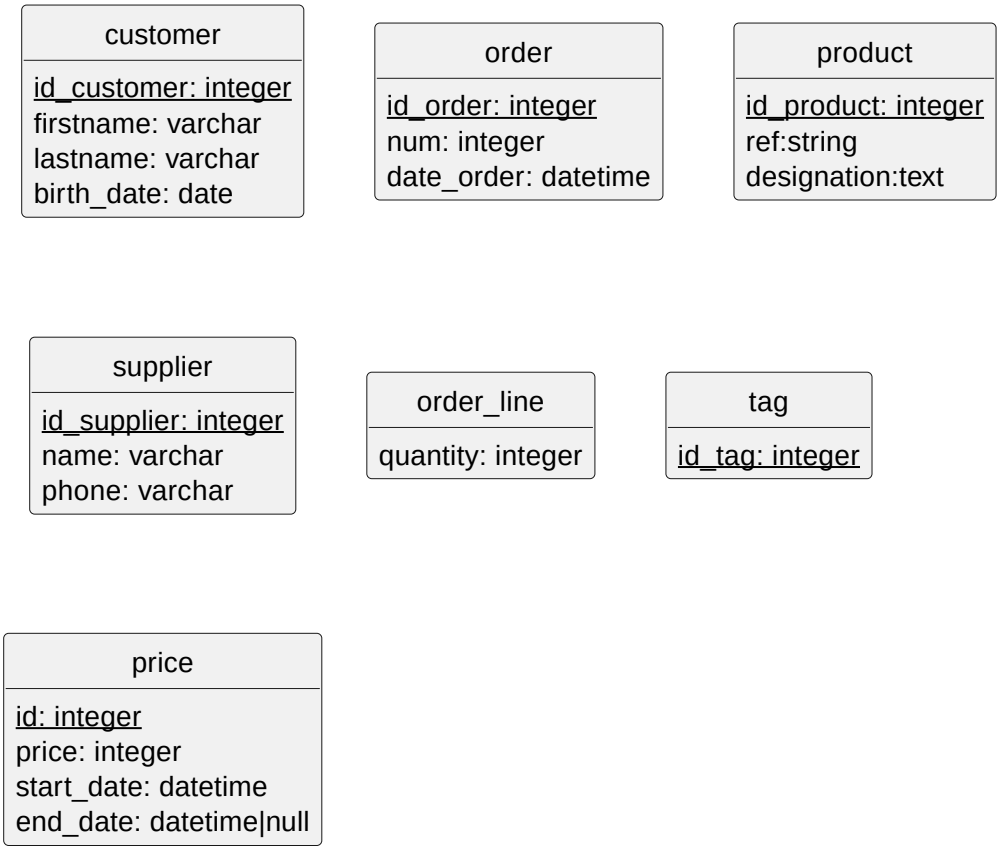


Sur le diagramme de classe, 7 classes sont marquées comme étant des "entités" (entity). Il y aura donc pour commencer 7 tables dans notre schéma relationnel.

Pour chaque entité, il faut lister les propriétés dont la valeur doit être persistée. Il peut arriver que certaines propriétés ne soient utiles que dans l'application sans qu'ils soient nécessaire de conserveur leur valeur dans une table. Dans ce cas, ces propriétés ne sont pas reprises dans le schéma relationnel. Nous, nous souhaitons toutes les conserver.

Les propriétés qui sont marquées avec `{id}` doivent être soulignées dans le schéma relationnel.

A ce niveau, voici l'avancé de notre schéma relationnel :

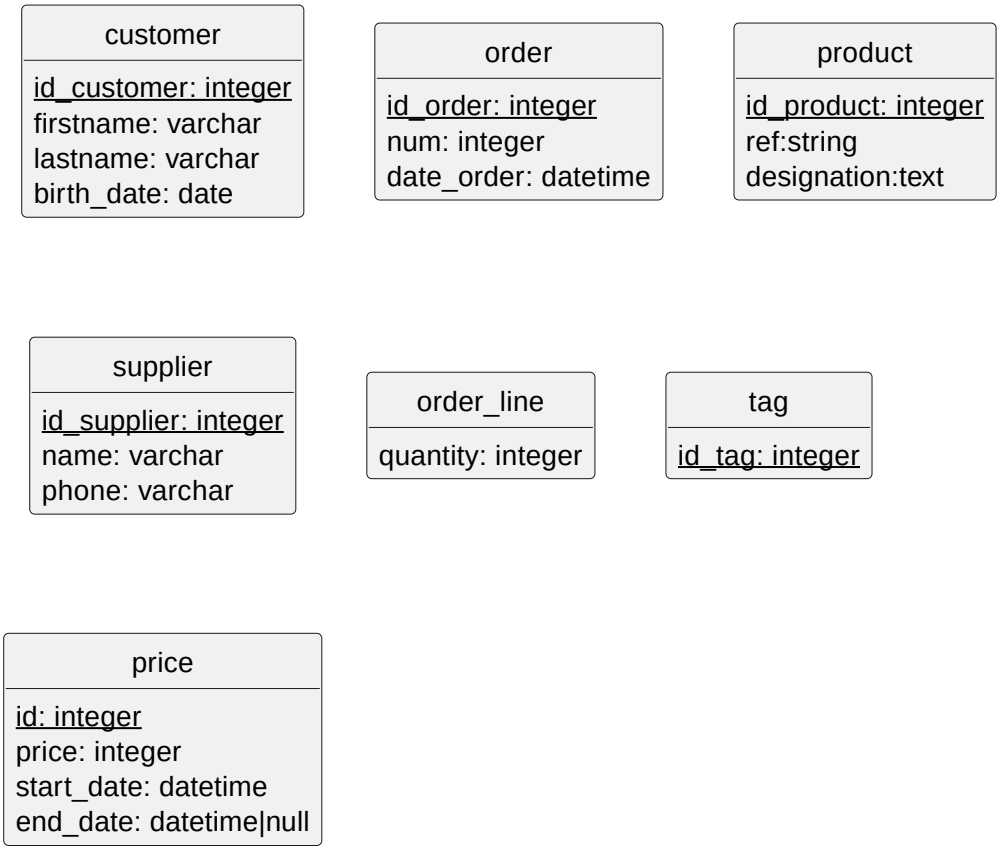


2.2.2. Identifier les relations entre les tables

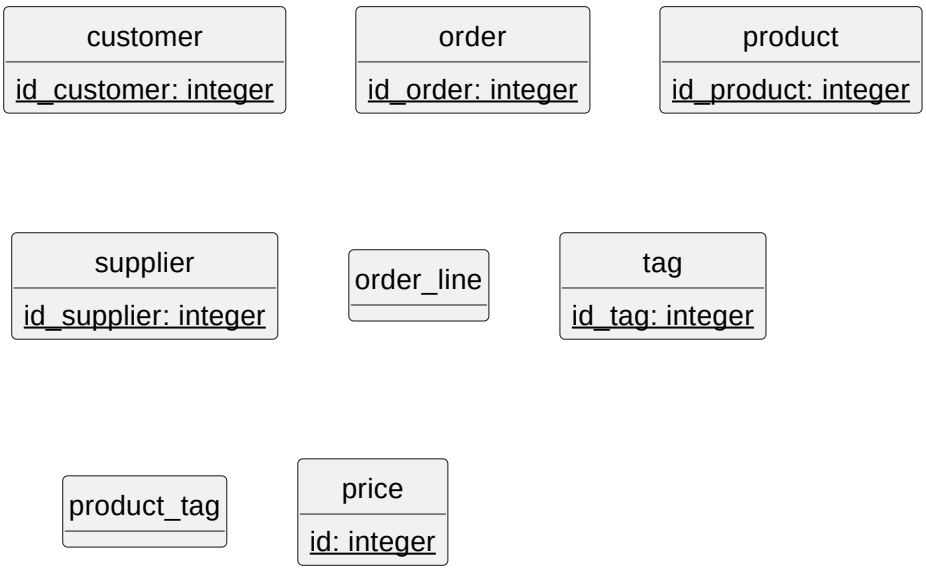
Nos tables sont isolées les unes des autres.

Les associations dans le diagramme de classe traduisent un lien.

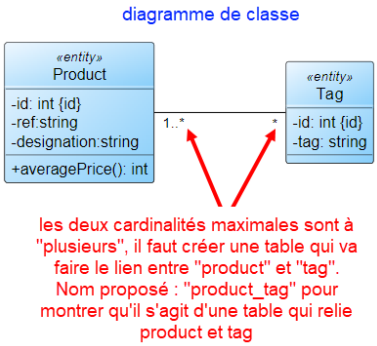
1. Seules les classes qui sont des **entités** doivent être conservées dans le schéma relationnel (une entité devient une table, y compris s'il s'agit d'une entité classe association). Nous avons 7 classes "entity" ce qui nous fait 7 tables :



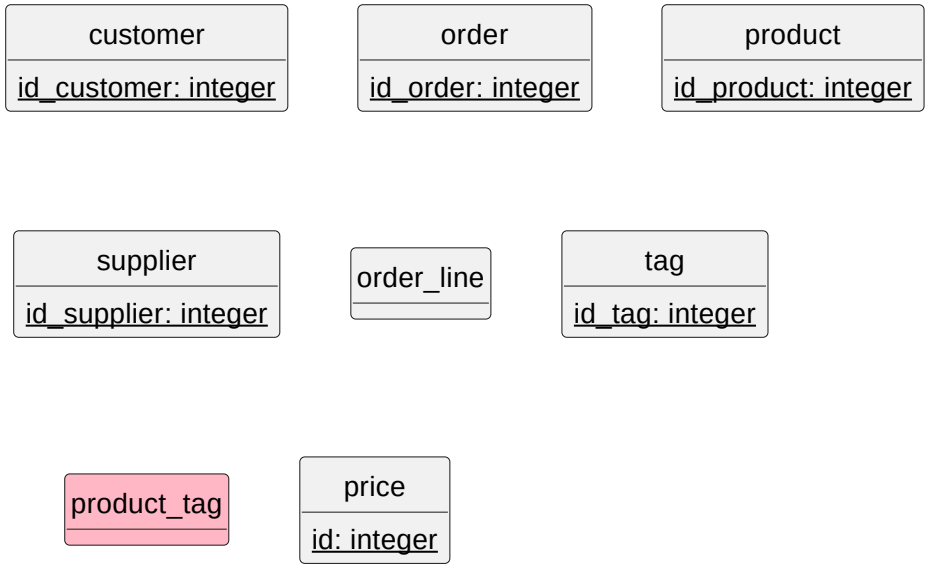
2. les propriétés marquées comme identifiant par `{id}` dans le diagramme de classe doivent être soulignées dans le schéma relationnel :



3. les associations plusieurs à plusieurs (**associations many to many**) qui n'ont pas de classe association donnent une table supplémentaire :



Cela nous donne une table supplémentaire `product_tag`



4. Repérer les associations "**un à plusieurs** (**association one to many)**" :

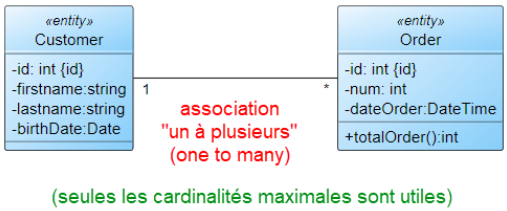
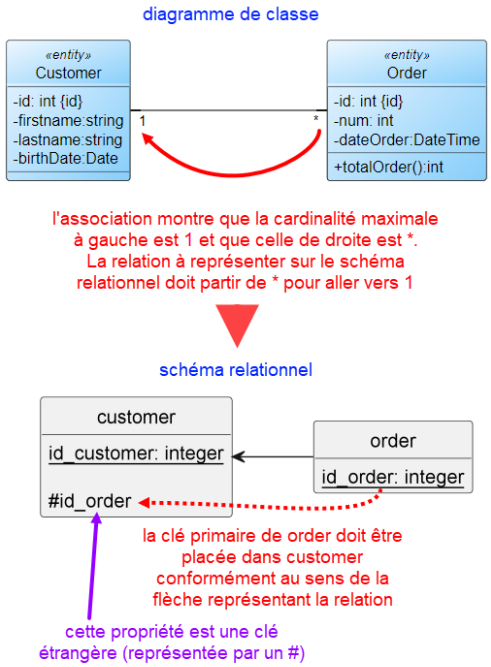


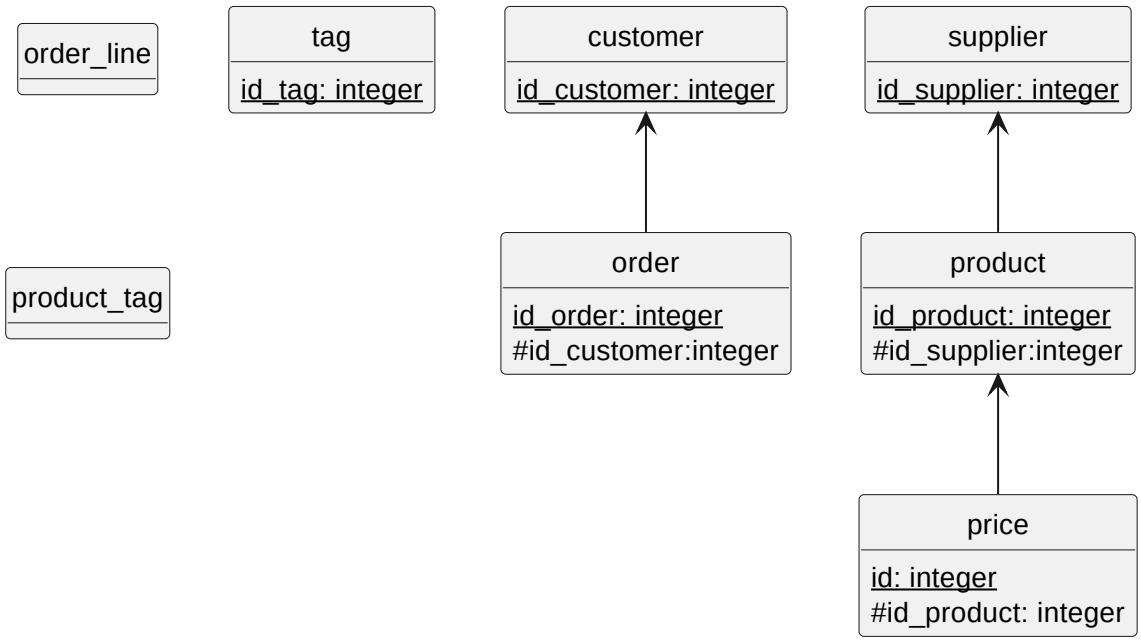
Figure 1. exemple d'association un à plusieurs (one to many)

Le sens de la flèche respecte une logique :

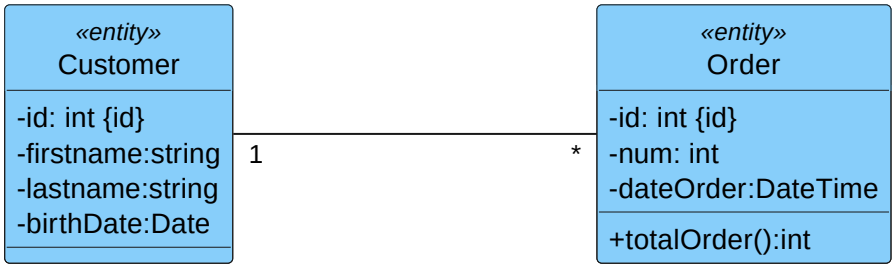
- une association avec cardinalités maximales * vers 1 donne une relation (flèche) qui part de celle avec * pour pointer celle avec 1 :



Après application de cette règle pour toutes les associations concernées, voici le résultat :



5. Repérer les associations "plusieurs à plusieurs" :



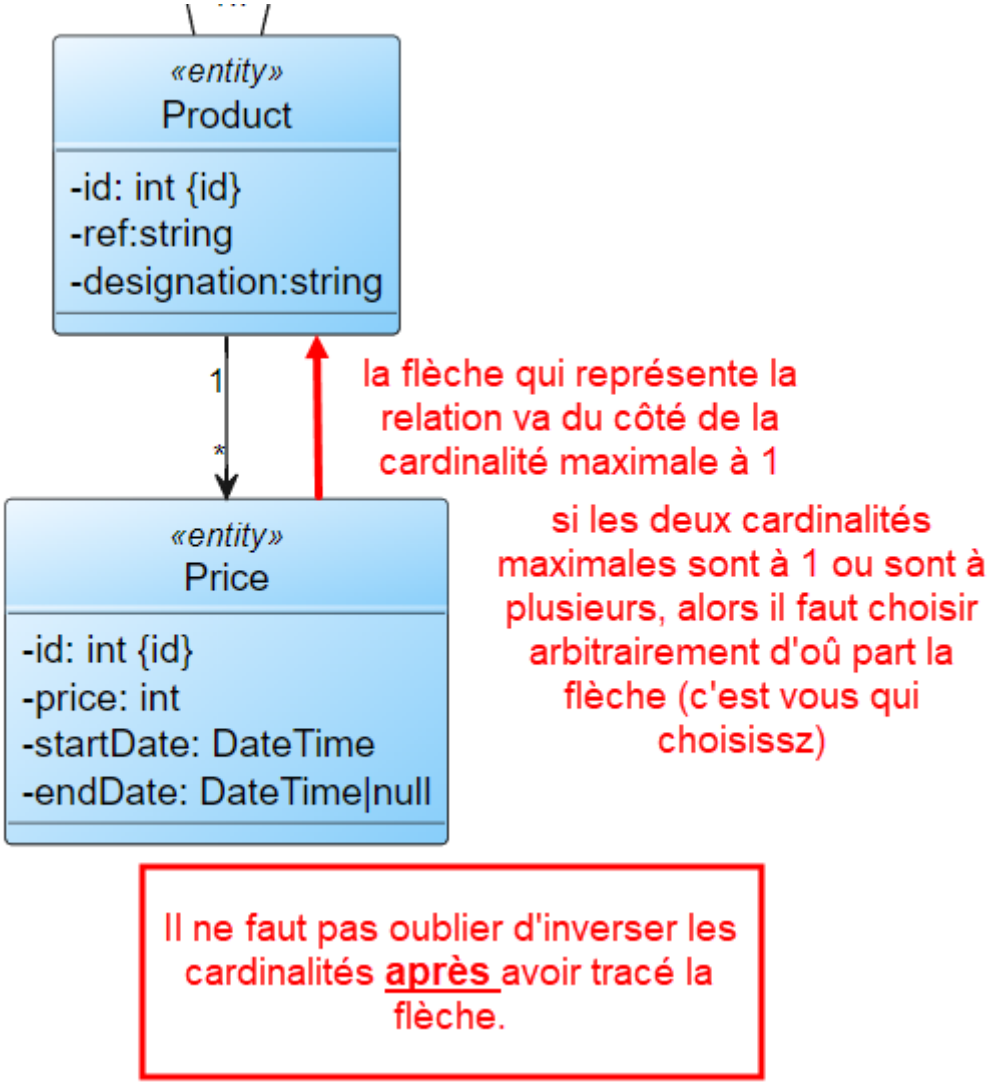
[] | *T:/SPOT_3WA/analyse/uml/diagramme-de-classes/document/assets/diagram_images/.svg*

- chaque entité devient une table (un rectangle sur le schéma)

- chaque classe association devient une [%autowidth]
- chaque association multiple (des deux côtés) devient une table "association"(le nom peut être constitué des deux tables séparés par un underscore) (sauf lorsqu'il y a une classe association (point précédent))
- le nom de chaque entité est écrit en minuscules et s'il' est noté en camelCase, il devra être noté en snake_case (car les SGBR sont souvent insensibles à la casse).
- une association devient une relation (trait qui se termine par une flèche entre deux tables).
    - La flèche pointe toujours l'entité ciblée par la cardinalité maximale à 1 (ex : entre Product et Supplier, la cardinalité maximale 1 est du côté de Supplier, la flèche partira de Product pour pointer Supplier)
    - si les deux entités ont une cardinalité maximale à 1, alors il faudra faire partir la flèche depuis l'entité dominante
    - si les deux entités ont une cardinalité maximale à plusieurs,
- l'identifiant d'une classe devient une clé primaire. Elle est soulignée dans le schéma

On rejoint l'analyse merise qui est une autre méthode d'analyse mais uniquement tournée base de données. L'objectif est de faciliter les échanges entre deux personnes, une qui utiliserait UML et l'autre Merise. Avec le schéma relationnel, elles "parlent" de la même chose (notamment au niveau des cardinalités)

- les méthodes n'apparaissent pas dans le schéma relationnel (tout simplement parce qu'elles ne concernent pas la bdd, seulement le côté applicatif)
- les relations sont représentées par une flèche :
    - une flèche par de l'entité qui a la cardinalité maximale à plusieurs pour aller vers l'entité qui a la cardinalité maximale à 1.
    - si les deux cardinalités maximales de la relation sont à 1, il faut choisir l'entité dominante.



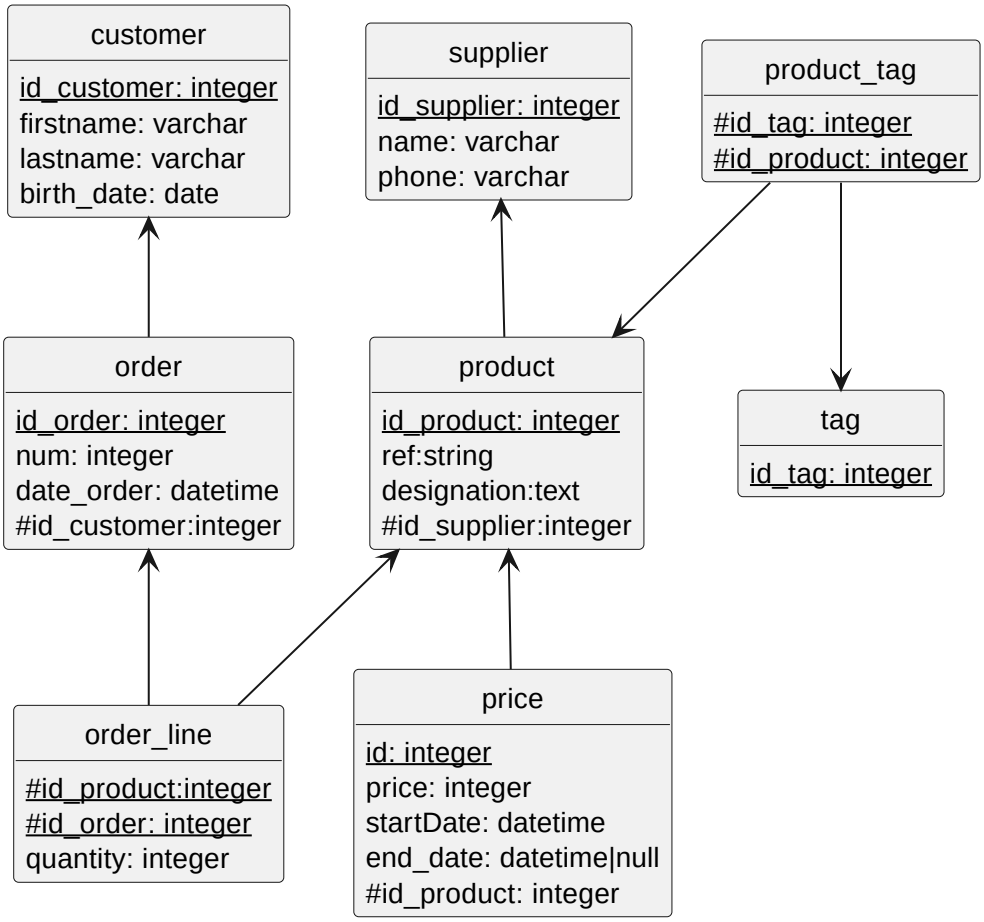
    - si les deux cardinalités maximales sont à multiples, l'association devient une classe à part entière.

- les cardinalités du diagramme UML doivent être inversées (après avoir respecté les étapes précédentes)
- lorsqu'il y a

La notion d'association unidirectionnelle / bidirectionnelle n'existe pas dans un schéma relationnel. Seul le sens du lien entre deux tables apparaît avec la flèche.

Pour faire le schéma relationnel, j'utilise plantuml et la syntaxe pour les diagrammes de classes. Mais le schéma relationnel n'est pas un diagramme de classe !

Je détourne seulement l'usage de plantuml pour faire le schéma relationnel.



Code source (qui détourne planuml pour faire un schéma relationnel)

```
hide circle
hide method
skinparam classAttributeIconSize 0
skinparam RectangleBackgroundColor white

class Customer {
{static} idCustomer: integer
}

class Order {
{static} idOrder: int
#idCustomer:integer
}

Customer "*" <-- "1" Order
```

[1] Le terme est abusif, il faudrait parler de relation puisque nous sommes dans un schéma relationnel. Mais pour faciliter la compréhension, j'utiliserai le terme de table.