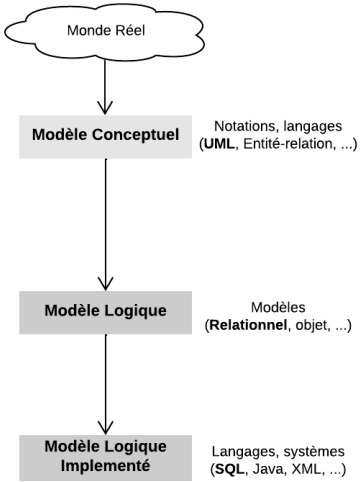
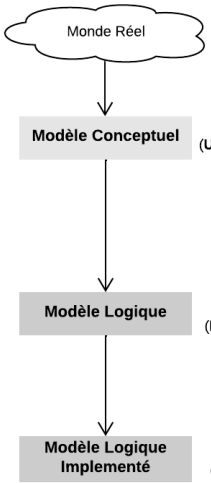
Chapitre 5 - Conception UML et transformation en modèle Relationnel

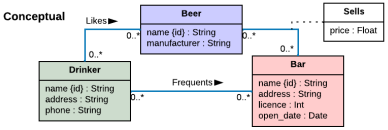
Processus de Conception



Processus de Conception



Conceptual



Logical

Beers (beer_name, beer_manufacturer)

Drinkers (drinker_name, drinker_address, drinker_phone)

Bars (bar_name, bar_address, bar_licence, bar_open_date)

Likes (drinker_name, beer_name)

Likes[drinker_name] ⊆ Drinkers[drinker_name]

Likes[beer_name] ⊆ Beers[beer_name]

Frequents (drinker_name, bar_name)

Frequents[drinker_name] ⊆ Drinkers[drinker_name]

Frequents[bar_name] ⊆ Bars[bar_name]

Sells (bar_name, beer_name, sells_price)

sells_price > 0

Sells[beer_name] ⊆ Beers[beer_name]

Sells[bar_name] ⊆ Bars[bar_name]

SQL Implementation

```
-- SQLite Syntax
CREATE TABLE Beers (
 beer_name TEXT NOT NULL,
 -- NOT NULL not necessary for PK attributes in some systems such as ORACLE
 beer_manufacturer TEXT,
 CONSTRAINT pk_beers_c0 PRIMARY KEY (beer_name)
);

CREATE TABLE Bars (
     bar_name TEXT NOT NULL,
     bar_address TEXT,
     bar_licence DATE,
     bar_open_date DATE,
     CONSTRAINT pk_bars_c0 PRIMARY KEY (bar_name)
);

CREATE TABLE Sells(
 bar_name TEXT NOT NULL,
 beer_name TEXT NOT NULL,
 sells_price REAL NOT NULL,
 CONSTRAINT pk_sells_c0 PRIMARY KEY (bar_name,beer_name),
 CONSTRAINT fk_sells_c1 FOREIGN KEY (bar_name) REFERENCES Bars(bar_name),
 CONSTRAINT fk_sells_c2 FOREIGN KEY (beer_name) REFERENCES Beers(beer_name),
```

Les principaux éléments du diagramme de classes UML

*Adaptation pour les BDs

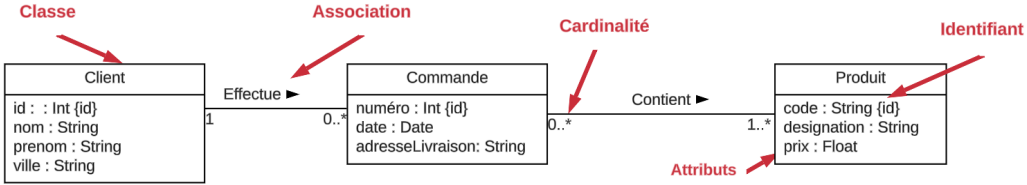
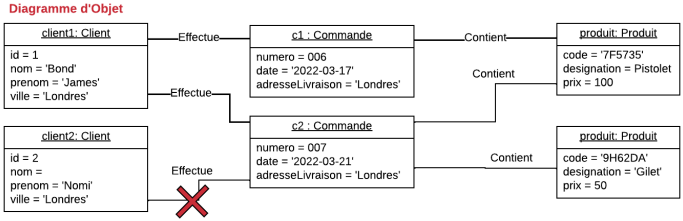


Diagramme d'Objets (instances des classes) et relations

*Adaptation pour les BDs



Clients (id_client, nom_client, prenom_client, ville_client)

id_client	nom_client	prenom_client	ville_client
1	Bond	James	Londres
2		Nomi	Londres

ProduitsCommandes (Contient)
(numero_commande, code_produit)

numero_commande	code_produit
006	7F5735
007	7F5735
007	9H62DA

Produits (code_produit, designation_produit, prix_produit)

code_produit	designation_produit	prix_produit
7F5735	Pistolet	100
9H62DA	Gilet	50

Commandes (numero_commande, date_commande, adresse_livraison_commande, id_client)

numero_commande	date_commande	adresse_livraison_commande	id_client
006	2022-03-17	Londres	1
007	2022-03-21	Londres	1

Modèle relationnel vs diagramme de classes UML (DB)

Le modèle relationnel et le diagramme de classes UML ont des correspondances évidentes:

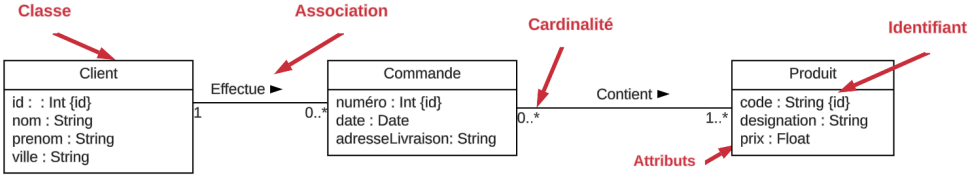
Modèle relationnel	*Diagramme de classes UML*
Attributes	Attributs
Schéma des relations	Classes et Associations
n-uplet	valuers des objets

Différence majeure:

1. Les relations sont utilisées pour modéliser à la fois les classes et les associations

Exemple UML vers modèle relationnel

UML (niveau conceptuel):



Modèle Relationnel (niveau logique):

Clients(id_client, nom_client, prenom_client, ville_client)
Commandes(numero_commande, date_commande, adresse_livraison_commande, id_client)
Produits(code_produit, designation_produit, prix_produit)
ProduitsCommandes(numero_commande, code_produit)*/* suite à l'asso many-to-many Contient */*
Commandes[id_client] $\subseteq$ Clients[id_client]
ProduitsCommandes[numero_commande] = Commandes[numero_commande]
/ égalité car toute commande apparaît dans la table ProduitsCommandes (1..*) */*
ProduitsCommandes[code_produit] $\subseteq$ Produits[code_produit]

Autres éléments du diagramme de classes UML

*Adaptation pour les BDs

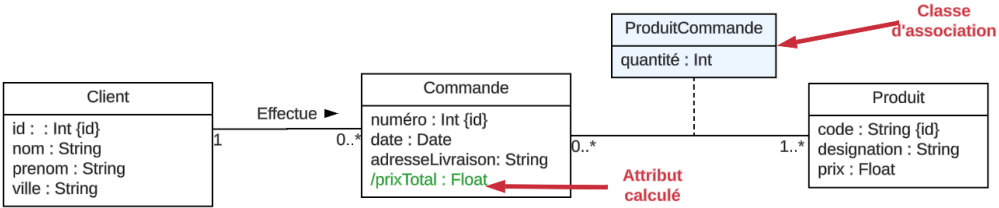
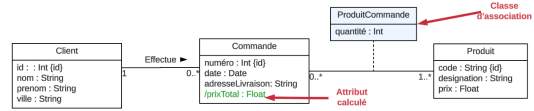
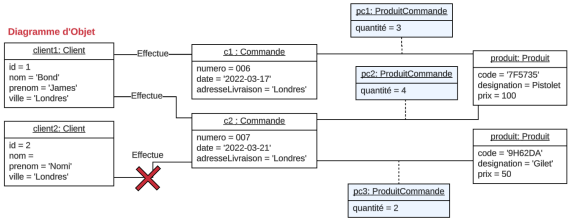


Diagramme d'Objets (instances des classes) et relations

**On traite la création des vues plus tard





Clients
(id_client, nom_client, prenom_client, ville_client)

id_client	nom_client	prenom_client	ville_client
1	Bond	James	Londres
2		Nomi	Londres

ProduitsCommandes
(numero_commande, code_produit, quantite_produitcommande)

numero_commande	code_produit	quantite_produitcommande
006	7F5735	3
007	7F5735	4
007	9H62DA	2

Produits
(code_produit, designation_produit, prix_produit)

code_produit	designation_produit	prix_produit
7F5735	Pistolet	100
9H62DA	Gilet	50

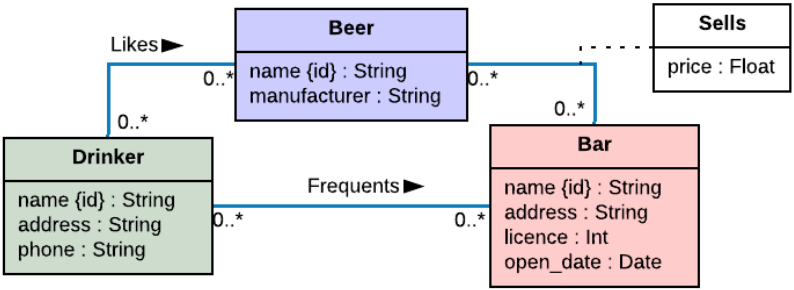
Commandes_base
(numero_commande, date_commande, adresse_livraison_commande, id_client)

numero_commande	date_commande	adresse_livraison_commande	id_client
006	2022-03-17	Londres	1
007	2022-03-21	Londres	1

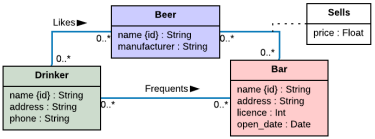
View Commandes
(numero_commande, date_commande, adresse_livraison_commande, id_client, prix_total_commande)

numero_commande	date_commande	adresse_livraison_commande	id_client	prix_total_commande
006	2022-03-17	Londres	1	300
007	2022-03-21	Londres	1	500

Exemple UML vers modèle relationnel



Exemple UML vers modèle relationnel



Drinkers (drinker_name, drinker_address, drinker_phone) */* drinker_name est la clé */*
Beers (beer_name, beer_manufacturer)
Bars (bar_name, bar_address, bar_license, bar_open_date)
Likes (drinker_name, beer_name) */* drinker_name, beer_name est la clé */*
Likes[drinker_name] $\subseteq$ Drinkers[drinker_name]
Likes[beer_name] $\subseteq$ Beers[beer_name]
Sells (bar_name, beer_name, sells_price)
domain(sells_price) $=$ integers > 0
Sells[beer_name] $\subseteq$ Beers[beer_name]
Sells[bar_name] $\subseteq$ Bars[bar_name]
Frequents (drinker_name, bar_name)
Frequents[drinker_name] $\subseteq$ Drinkers[drinker_name]
Frequents[bar_name] $\subseteq$ Bars[bar_name]

Tranformation de l'UML vers Modèle Relationnel : Objectifs

La mise en œuvre d'une application basée sur un SGBD doit transformer la modèle conceptuel en modèle relationnel.
Les objectifs sont:

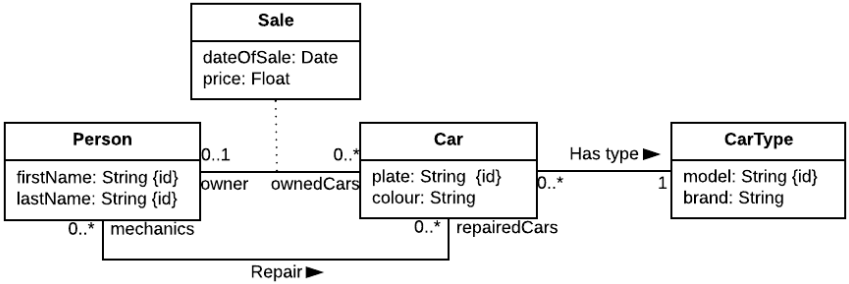
- **Représenter toutes les informations** contenues dans un diagramme de classes (classes, associations et contraintes)
- **Éviter la redondance**
  (produire un schéma relationnel de bonne qualité)
- **Fournir un schéma relationnel aussi simple que possible**
  (évitez trop de relations)
- **Éviter les valeurs absents**

Les principaux éléments UML: Exercice Garage

- Une voiture est identifiée par une plaque d'immatriculation. Elle est toujours attachée à un seul modèle et à une marque. Le modèle est unique et détermine la marque (ex. nous pouvons avoir 10 Renault Megane avec des plaques et des couleurs différentes).
- Une voiture appartient à une seule personne (le propriétaire) après une vente (*vente*). Une personne est identifiée par un prénom et un nom. Une personne peut posséder plusieurs voitures. La vente est capturée via une date ainsi que par le prix.
- Une voiture peut être réparée par plusieurs personnes (mécaniciens), et un mécanicien peut réparer plusieurs voitures.

Les principaux éléments UML: Exercice Garage

- Une voiture est identifiée par une plaque d'immatriculation. Elle est toujours attachée à un seul modèle et à une marque. Le modèle est unique et détermine la marque (ex. nous pouvons avoir 10 Renault Megane avec des plaques et des couleurs différentes).
- Une voiture appartient à une seule personne (le propriétaire) après une vente (*vente*). Une personne est identifiée par un prénom et un nom. Une personne peut posséder plusieurs voitures. La vente est capturée via une date ainsi que par le prix.
- Une voiture peut être réparée par plusieurs personnes (mécaniciens), et un mécanicien peut réparer plusieurs voitures.

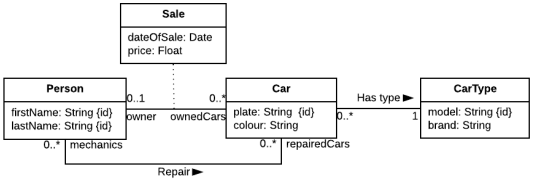


Chapitre 5 - Conception UML et transformation en modèle Relationnel

"Mapping" des Classes

Pour les classes qui ne sont pas classes d'association:

- Une relation pour chaque classe
- Les attributs des relations correspondent aux des attributs des classes
- Les éléments d'identification font partie d'une clé composée



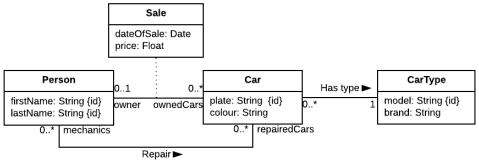
CarTypes(<u>model</u>, brand, ...
Cars(<u>plate</u>, colour, ...
Persons(<u>first_name, last_name</u>, ...

Chapitre 5 - Conception UML et transformation en modèle Relationnel

"Mapping" Associations (1 to Many)

Pour les associations 1 to Many (*):

- Ajouter des attributs dans la classe * faisant référence à la clé de la classe de cardinalité 1
- Comme la cardinalité est 1, le nouvel attribut (par exemple, *model*) ne peut pas être absent (not null)

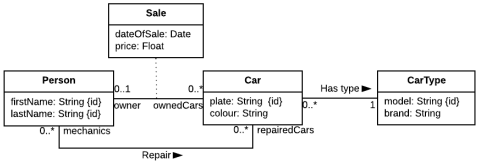


CarTypes(<u>model,</u> brand)
Cars(<u>plate</u>, colour, **model**)
Persons(<u>first_name, last_name</u>)
...
Cars[model] $\subseteq$ CarTypes[model]
model not null dans Cars

"Mapping" Associations (0..1 to Many)
Option 1 - On accepte des valeurs absents (NULL)

Pour les associations 0..1 à plusieurs (*), première option:

- Ajouter des attributs dans la classe * faisant référence à la clé de la classe de cardinalité 0..1 (admet les valeurs absents).



CarTypes(<u>model</u>, brand)
Cars(<u>plate</u>, colour, model, **owner_fn, owner_ln, date_of_sale, price**
Persons(<u>first_name, last_name</u>)
...

Cars[owner_fn, owner_ln] $\subseteq$ Persons[first_name, last_name]

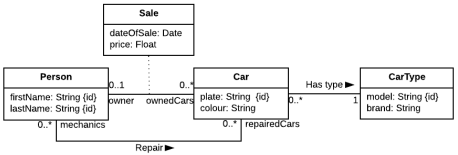
owner_fn, owner_ln, date_of_sale, price peuvent être absents (à cause du 0..1)

* fn = first name; ln = last name

"Mapping" Associations (0..1 to Many)
Option 2 (**préférable**) - On n'accepte pas des valeurs absents (NULL)

Pour les associations 0..1 à plusieurs (*) deuxième option:

- Une nouvelle relation où la clé de la classe de rôle 0 .. * est la clé de la relation créée.



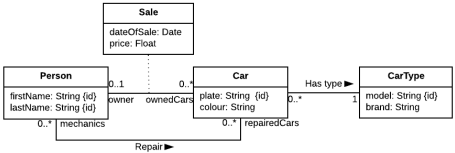
CarTypes(<u>model</u>, brand)
Cars(<u>plate</u>, colour, model)
Sales(<u>plate</u>, owner_fn, owner_ln, date_of_sale, price)
Persons(<u>first_name, last_name</u>)
...

Sales[plate] $\subseteq$ Cars[plate]
Sales[owner_fn, owner_ln] $\subseteq$ Persons[first_name, last_name];
owner_fn, owner_ln, plate, date_of_sale, price ne peuvent pas être absents;
* fn = first name; ln = last name

"Mapping" des Associations (Many to Many)

Pour les associations de plusieurs à plusieurs (*Many to Many*):

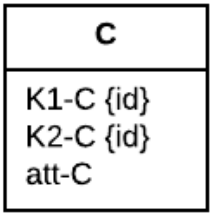
- Une nouvelle relation représente l'association.
- La **clé** de la nouvelle relation est **composée** d'attributs qui référencent les clés des relations associées.



CarTypes(<u>model</u>, brand)
Cars(<u>plate</u>, colour, model)
Sales(<u>plate</u>, owner_fn, owner_ln, date_of_sale, price)
Persons(<u>first_name, last_name</u>)
Reparations(<u>mechanics_fn, mechanics_ln, plate</u>)

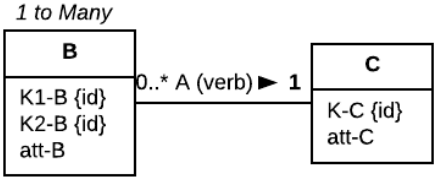
Reparations[mechanics_fn, mechanics_ln] $\subseteq$ Persons[first_name, last_name];
Reparations[plate] $\subseteq$ Cars[plate]; * fn = first name; ln = last name

Question 1



A. C(<u>K1-C</u>, K2-C, att-C)
B. C(<u>K1-C</u>, <u>K2-C</u>, att-C)
C. C(<u>K1-C, K2-C</u>, att-C)

Question 2 (Classe B)

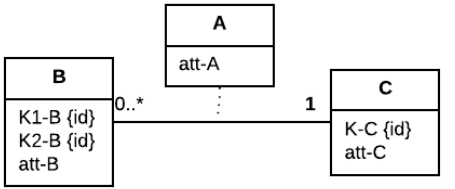


A. B(K1-B, K2-B, att-B)

B. B(K1-B, K2-B, att-B, K-C)

C. B(K1-B, K2-B, K-C, att-B)

Question 3 (Classes A et B)

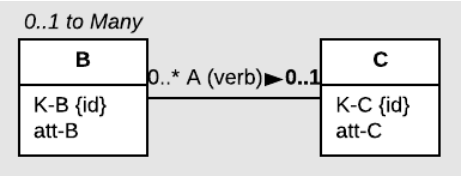


A. B(<u>K1-B, K2-B</u>, att-B, K-C, att-A)

B. B(<u>K1-B, K2-B</u>, att-B, K-C); A(<u>att-A</u>)

C. B(<u>K1-B, K2-B</u>, att-B); A(<u>K1-B, K2-B, K-C</u>, att-A)

Question 4 : (Classe B et Association A)



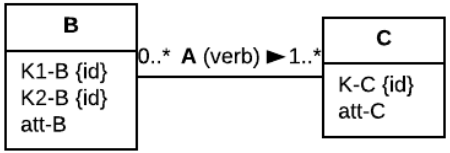
A. B(<u>K-B</u>, att-B); A(<u>K-B</u>, K-C)

B. B(<u>K-B</u>, att-B); A(K-B, <u>K-C</u>)

C. B(<u>K-B</u>, att-B, K-C)

Question 5 : Classes B et C et association A

Many to Many

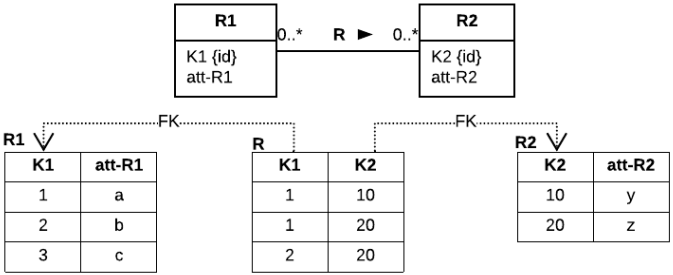


A. B(<u>K1-B, K2-B</u>, att-B, <u>K-C</u>, att-C)

B. B(<u>K1-B, K2-B</u>, att-B); A(<u>K1-B, K2-B</u>,K-C); C(<u>K-C</u>, att-C)

C. B(<u>K1-B, K2-B</u>, att-B); A(<u>K1-B, K2-B, K-C</u>); C(<u>K-C</u>, att-C)

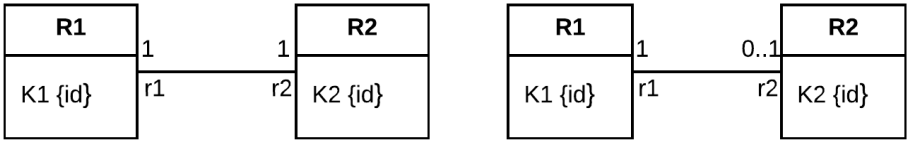
Rappel des règles avec associations plusieurs à plusieurs



Une nouvelle relation R est créée (liée à l'association elle-même):

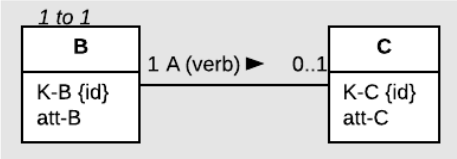
- K1 clé en R1, K2 clé en R2
- Attributs en R: K1 $\cup$ K2
- Clé en R: K1 $\cup$ K2
- Contraintes d'intégrité référentielle:
  R[K1] $\subseteq$ R1[K1] et R[K2] $\subseteq$ R2[K2]

"Mapping" associations (0..1 ou 1 dans les deux rôles)



- 1—— 1. 2 options :
  1. K1 devient attribut de R2 et R2[K1] = R1[K1], K1 est *autre* clé en R2.
  2. K2 devient attribut en R1 et R1[K2] = R2[K2], K2 est *autre* clé en R1
- 1 —— 0..1. Afin d'éviter des valeurs absents, un seul option:
  1. K1 devient attribut de R2 et R2[K1] $\subseteq$ R1[K1], K1 est *autre* clé de R2.

Question 7 : Classes B et C et association A



A. B(K-B, att-B); C(K-C, att-C, K-B)

B. B(K-B, att-B); C(K-C, att-C, K-B)

C. B(K-B, att-B); A(K-B,K-C); C(K-C, att-C)

Chapitre 5 - Conception UML et transformation en modèle Relationnel

Remarques finales

- Nous avons vu qu'une partie des possibilité de conception avec UML
- Il n'y a pas de théorème, rien d'autre que du bon sens!
- Si quelque chose ne va pas avec vos relations, revenez à la modélisation conceptuelle. Ne corrigez pas les relations!
- Si on construit un "bon" modèle conceptuel, cela impliquera une bonne qualité des relations après la traduction (avec peu de redondances).
- Parfois, il faut un peu de nettoyage: vérifiez les relations "inutiles" (incluses dans une autre).