Exemple d'utilisation des diagrammes de classes : réservations de vols

François Schwarzentruber
ENS Cachan – Antenne de Bretagne

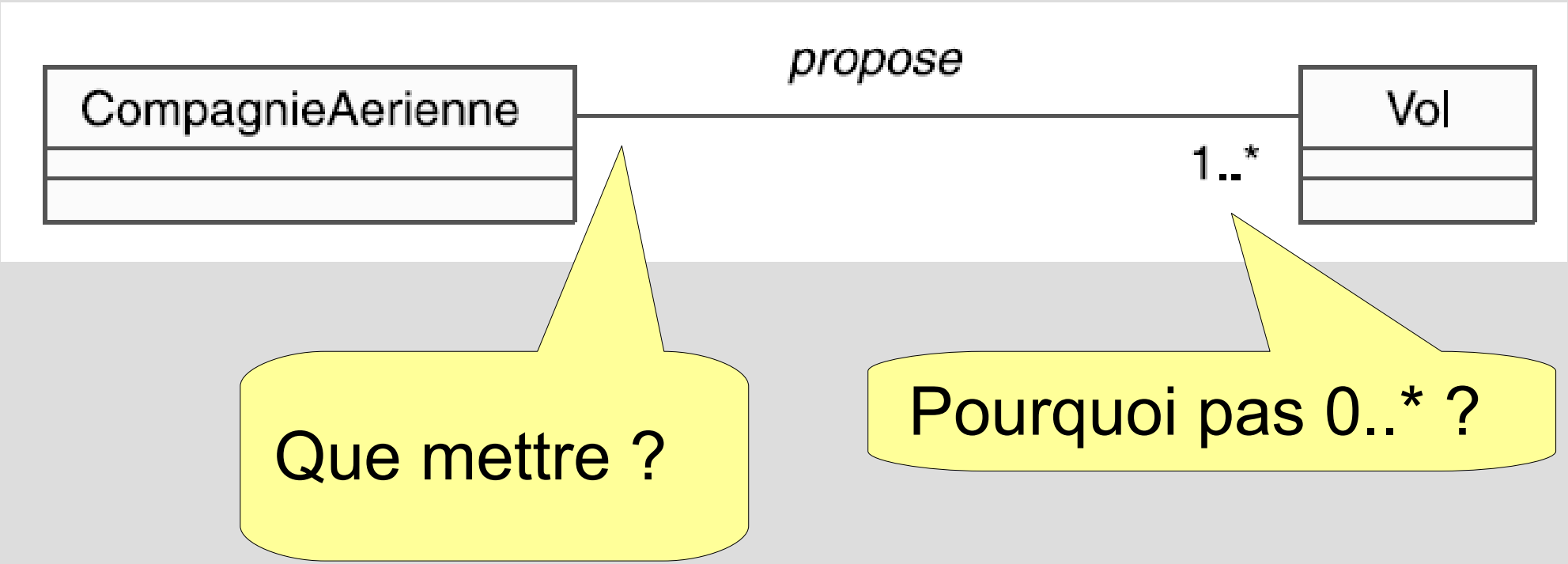
Référence : Pascal Roques. UML2 par la pratique, 6e édition.

Plan

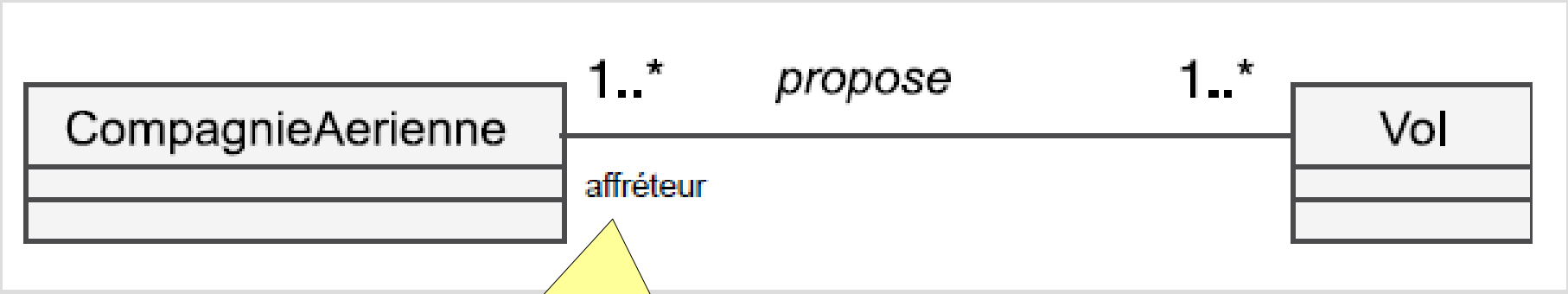
- Modélisation simple
- Attributs VS Classes
- Vers une classe d'association
- Quelques fioritures
- Forte cohésion
- Découpe en packages

Modélisation simple

Des compagnies aériennes proposent différents vols.

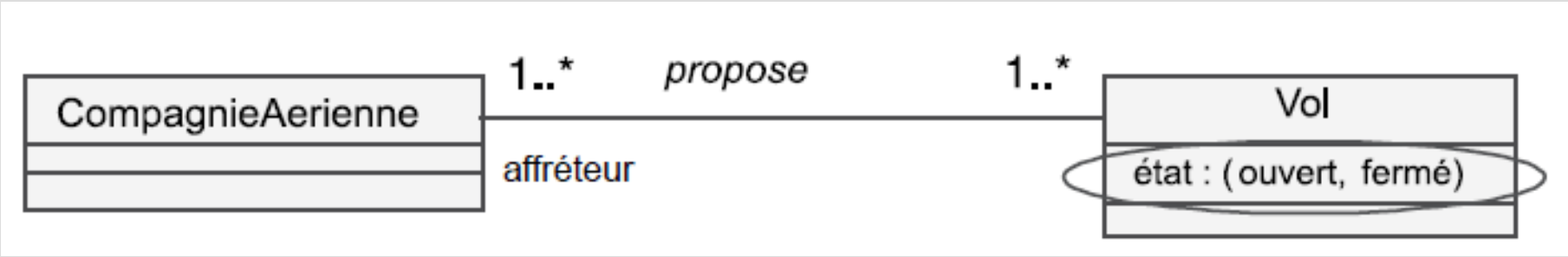


Des compagnies aériennes proposent différents vols.

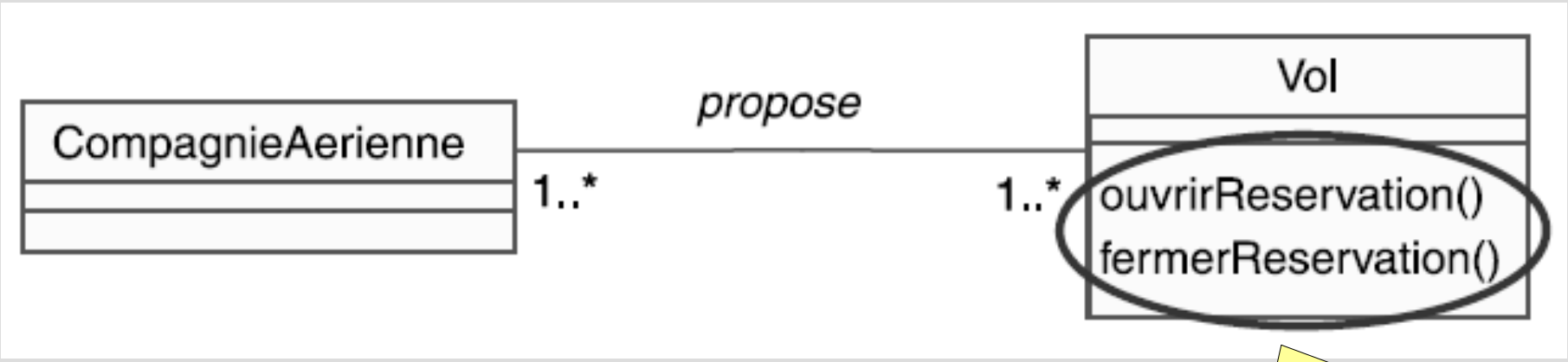


Ça arrive qu'il y ait plusieurs affréteurs !

Un vol est ouvert à la réservation et fermé sur ordre de la compagnie.



Un vol est ouvert à la réservation et fermé sur ordre de la compagnie.



On met les opérations là où elles s'exécutent...

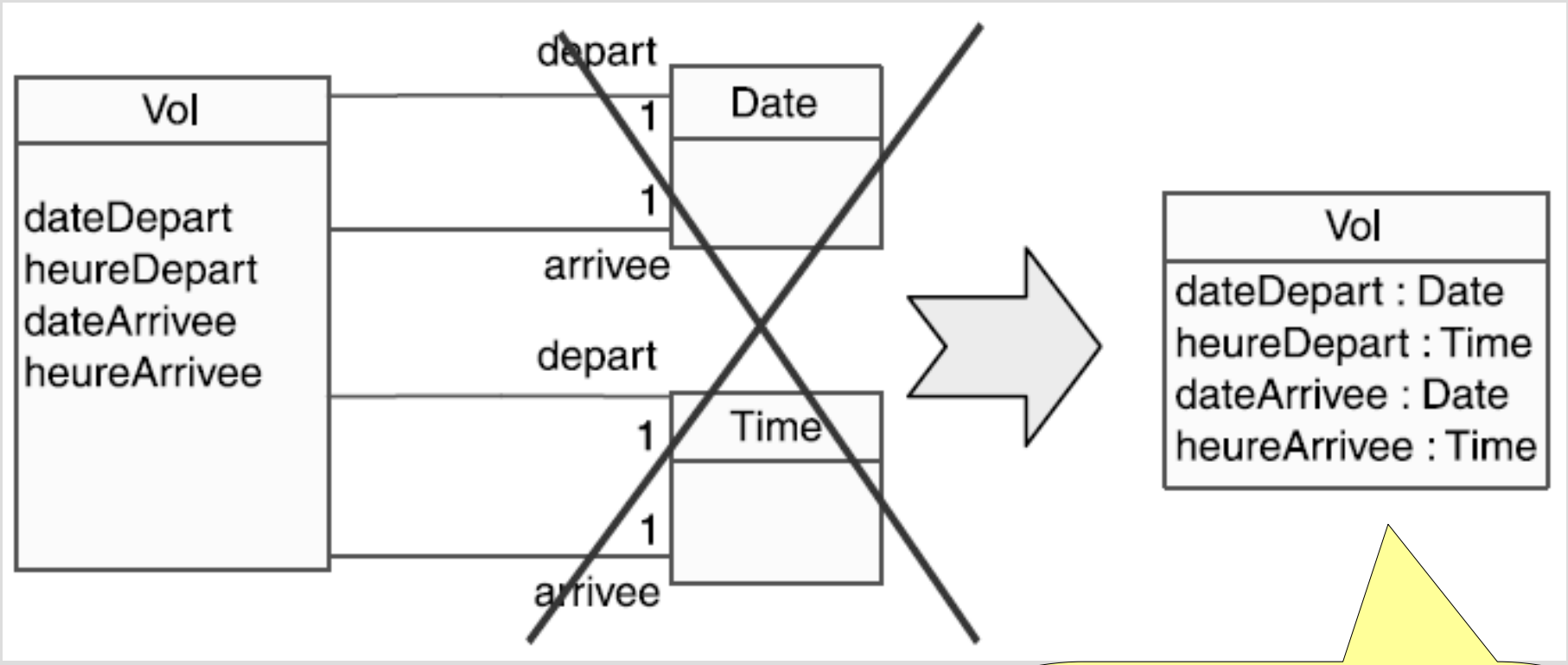
Même si c'est elle qui commande !

Attributs ou classes ?

Un vol a un jour et une heure de départ et un jour et une heure d'arrivée.

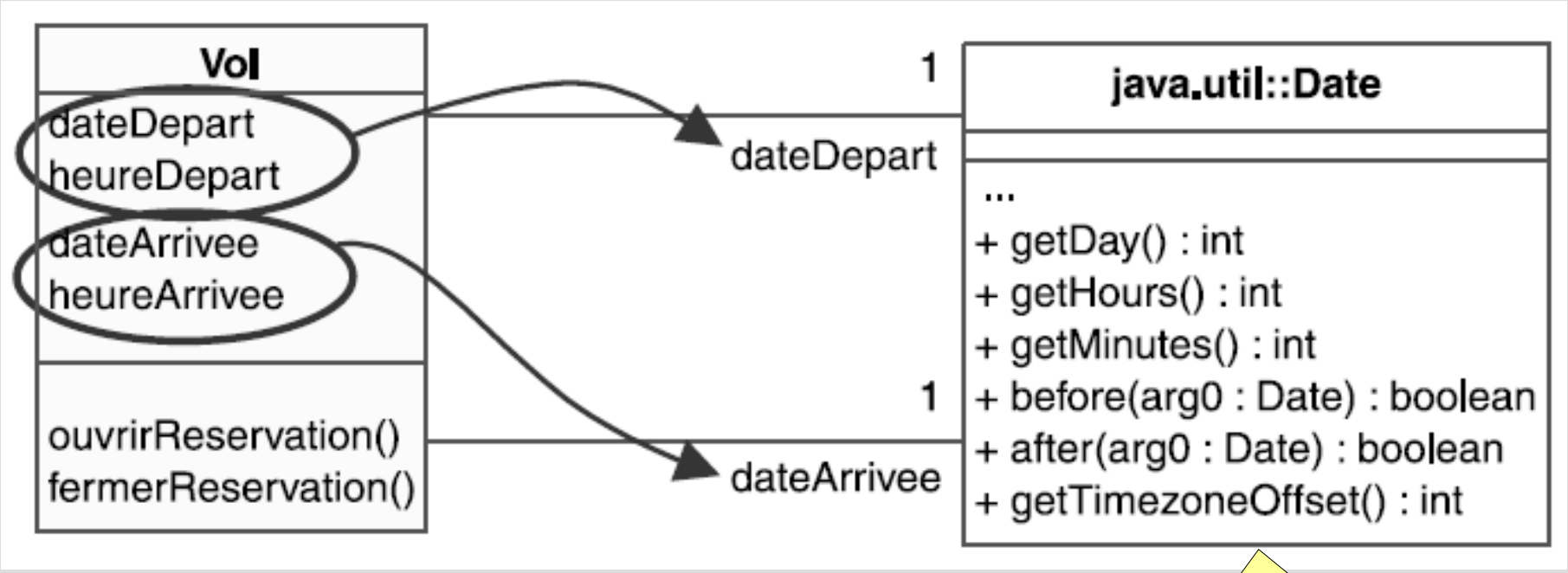
On crée des classes « Heure » et « Jour » ?

Un vol a un jour et une heure de départ et un jour et une heure d'arrivée.



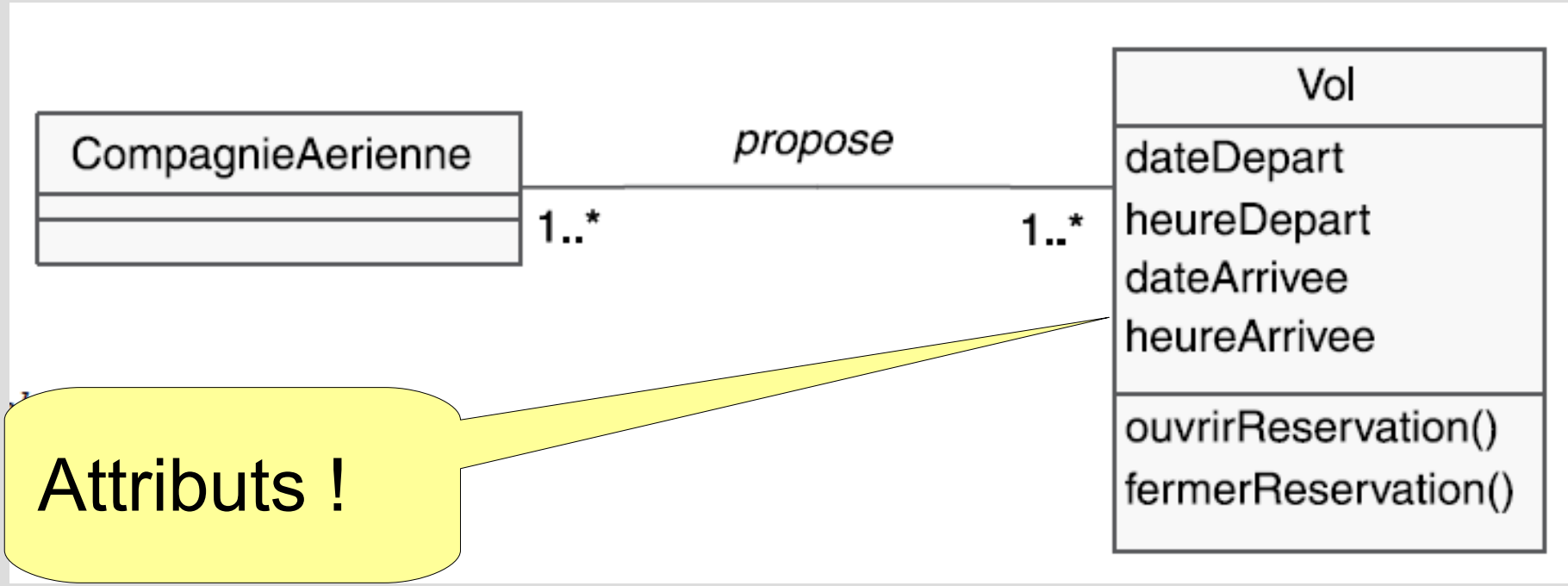
Non... car ce sont juste des valeurs...

Un vol a un jour et une heure de départ et un jour et une heure d'arrivée.



Moui... mais là on rentre trop dans le détail ! De l'abstraction !

Un vol a un jour et une heure de départ et un jour et une heure d'arrivée.



Un vol a un aéroport de départ et un aéroport d'arrivée.

Attributs !

On crée une classe « Aéroport » ?

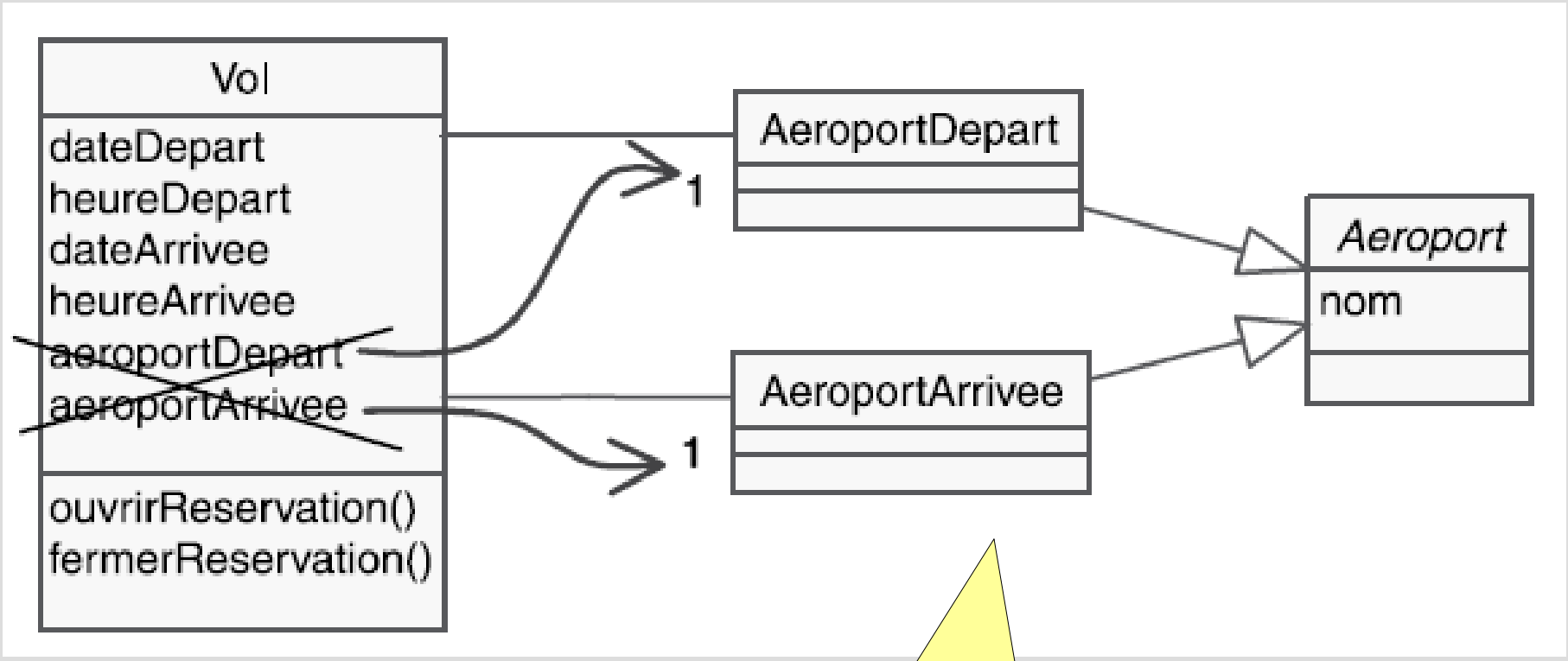
On crée des classes « Aéroport de départ » et « Aéroport d'arrivée » ?

Un vol a un aéroport de départ et un aéroport d'arrivée.

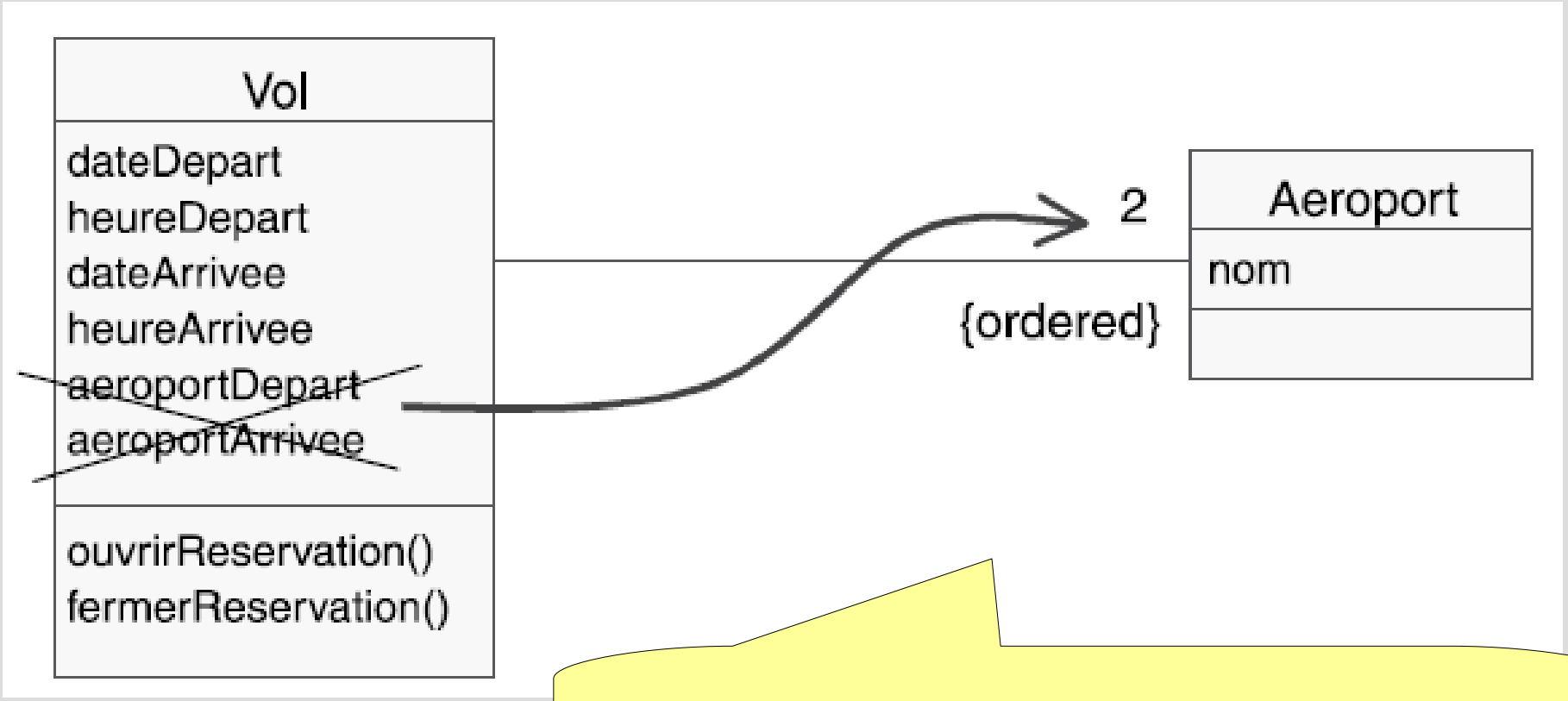
Attributs !

Non, car les aéroports ont des rôles :
- ils desservent des villes
- etc.

Un vol a un aéroport de départ et un aéroport d'arrivée.

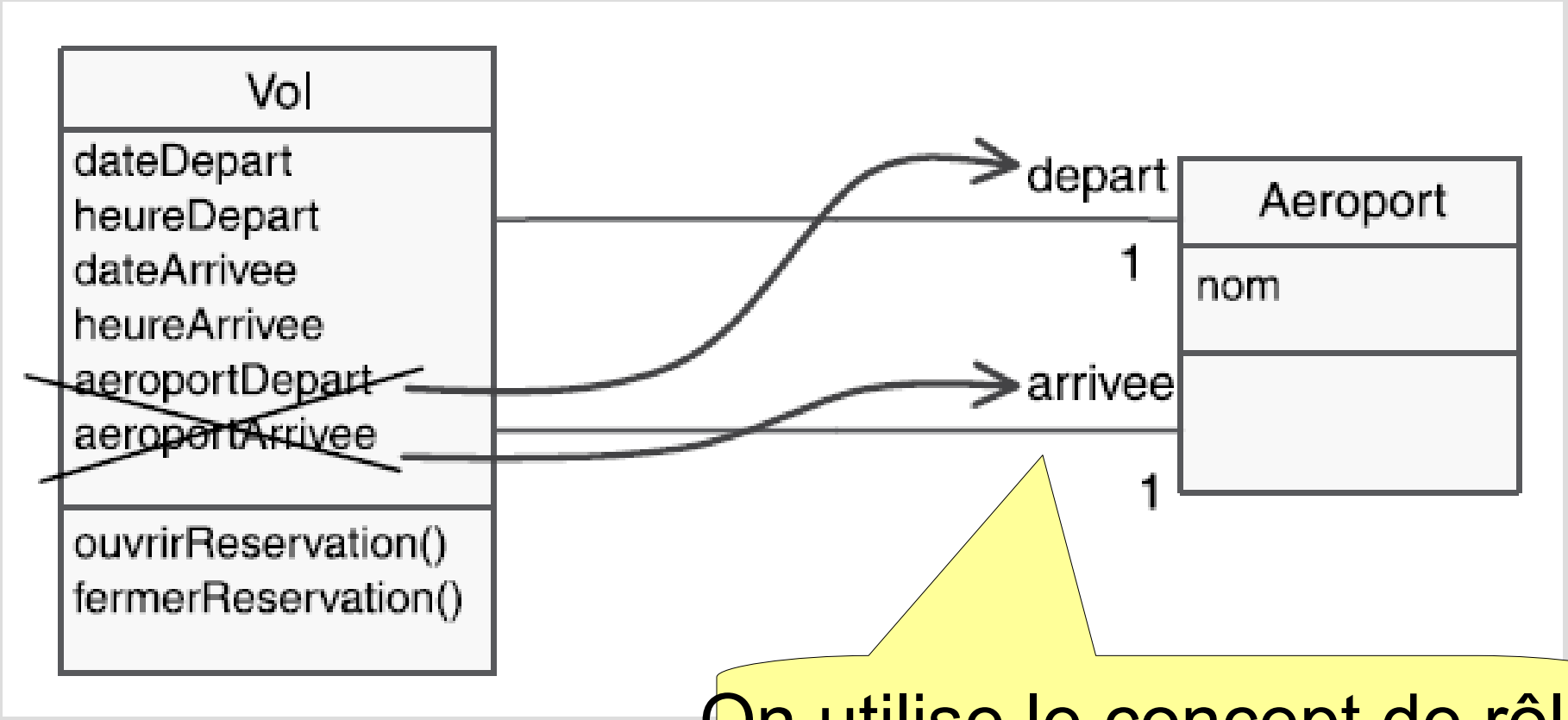


Un vol a un aéroport de départ et un aéroport d'arrivée.



Pas stupide mais... difficile à lire...

Un vol a un aéroport de départ et un aéroport d'arrivée.



On utilise le concept de rôle !

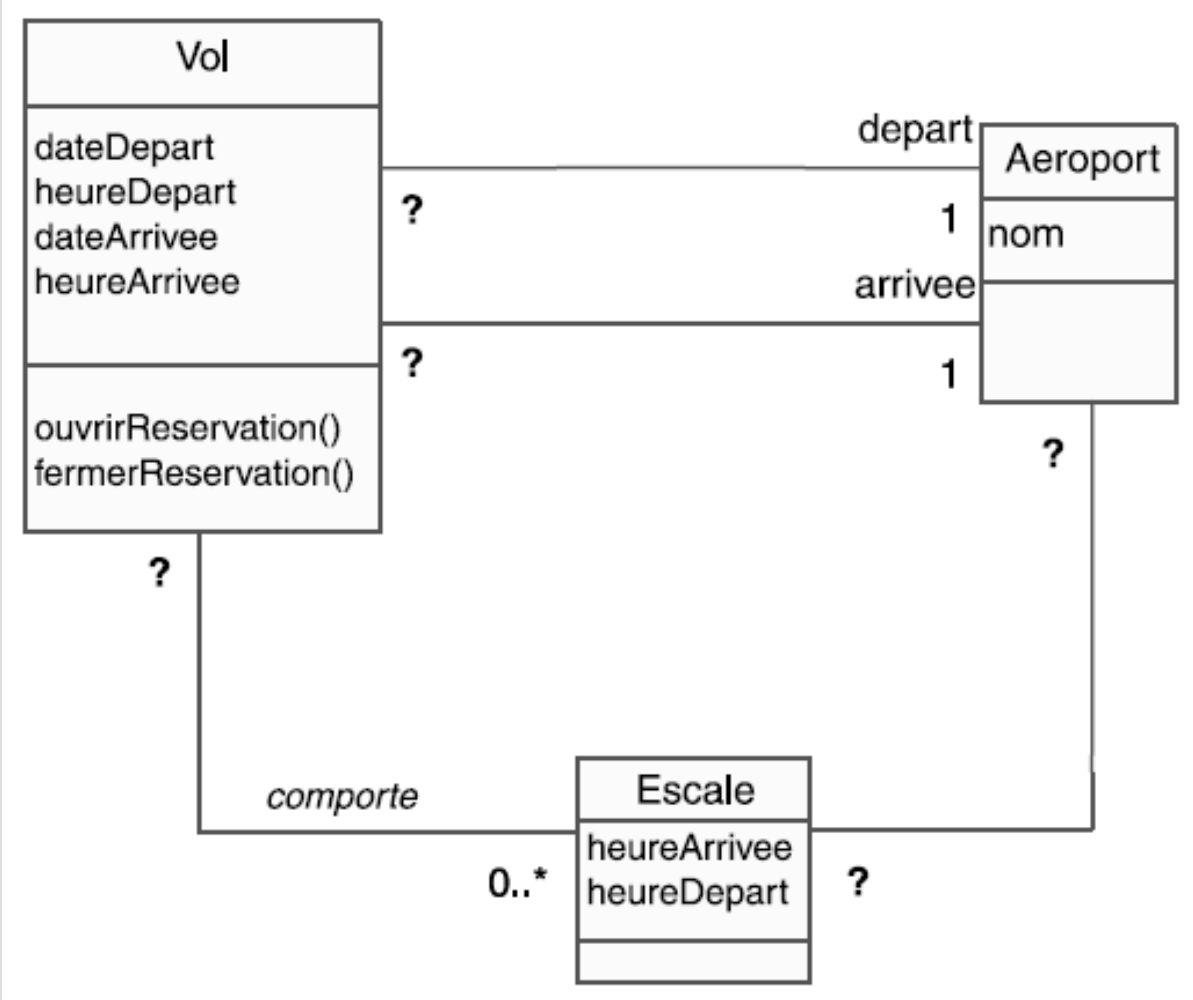
Chaque aéroport dessert une ou plusieurs villes.

Chaque aéroport dessert une ou plusieurs villes.

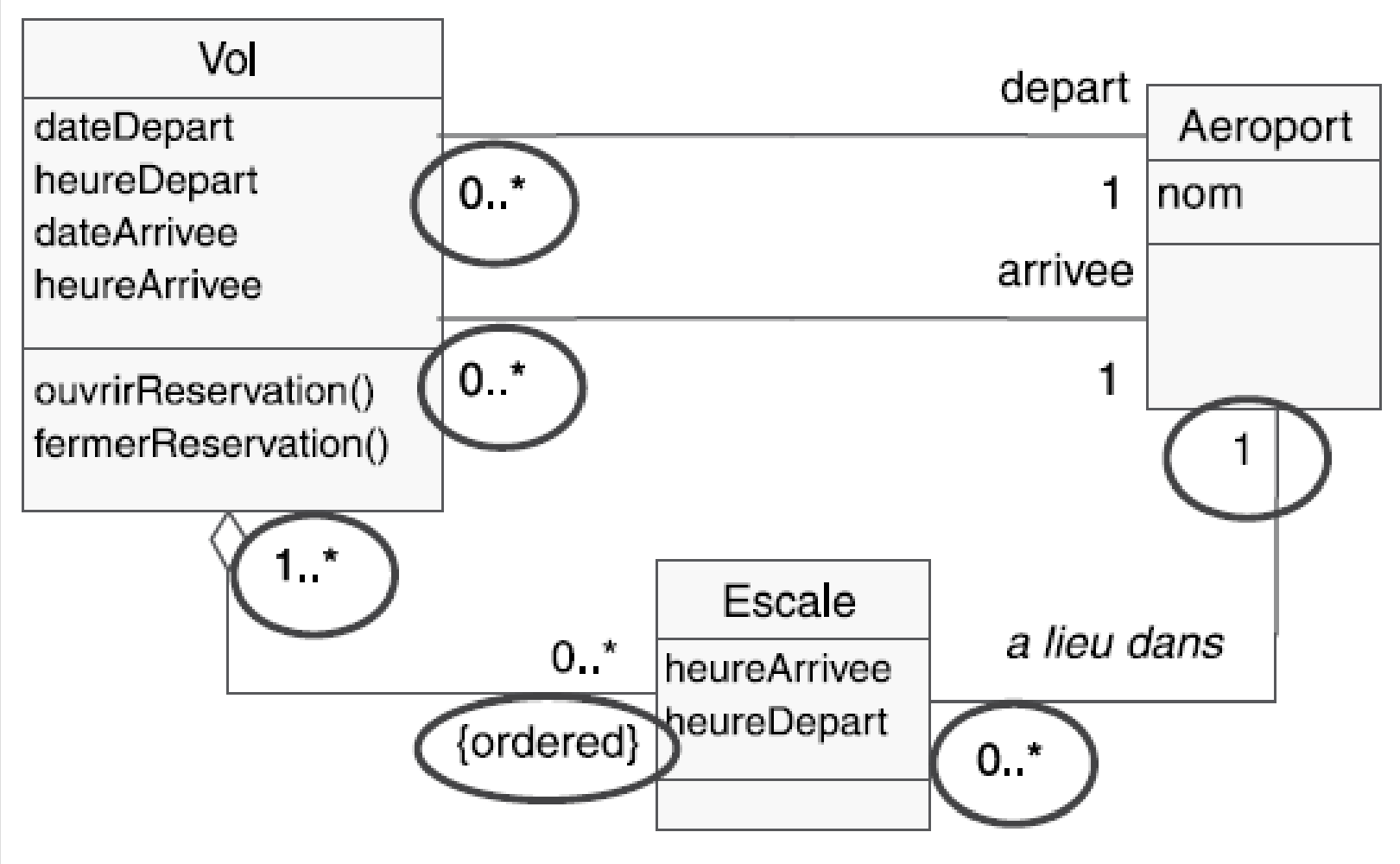
Bon… faisons ce choix.

Vers une classe d'association

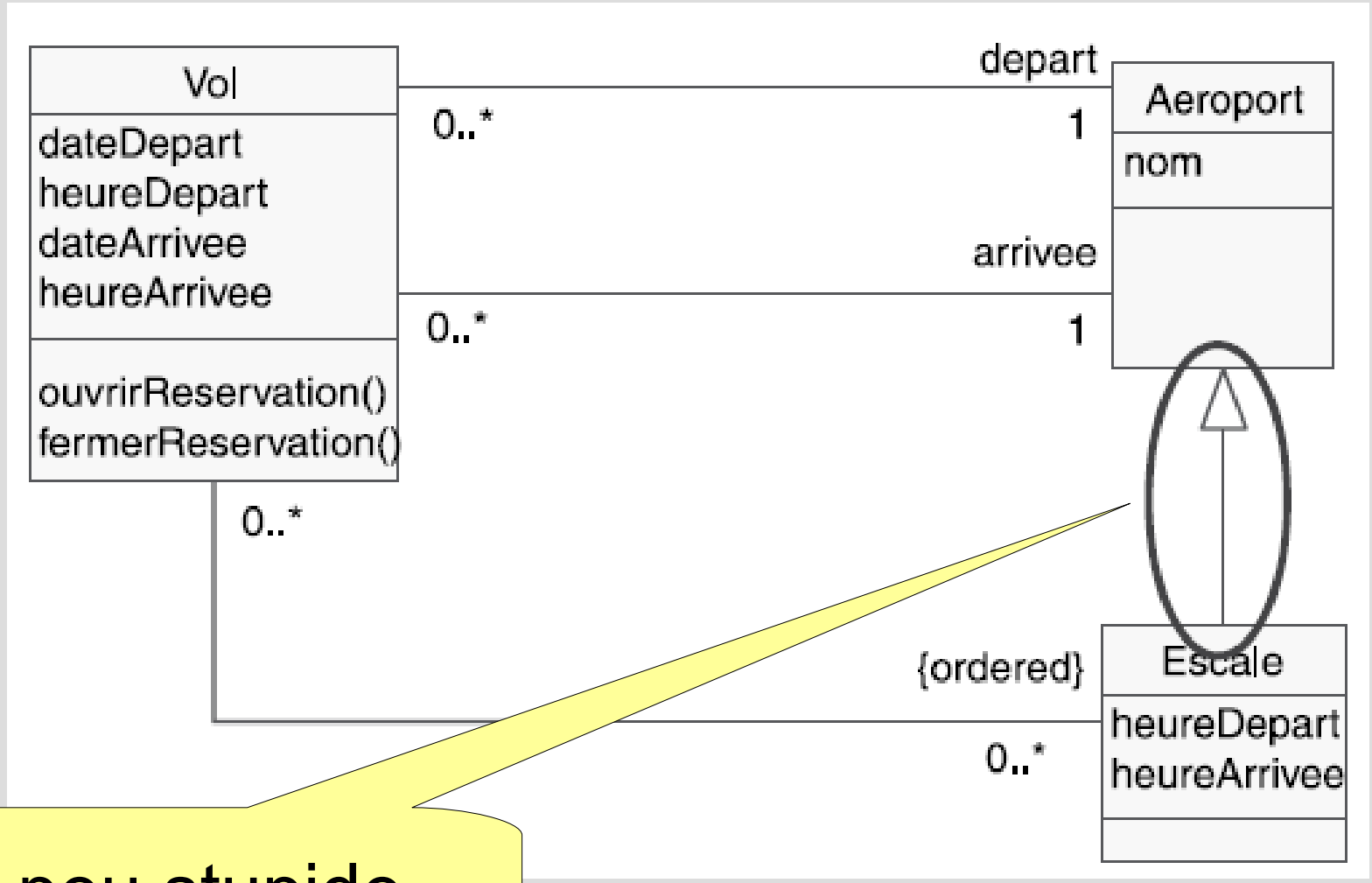
Un vol peut comporter des escales dans des aéroports.
Une escale a une heure d'arrivée et une heure de départ.



Un vol peut comporter des escales dans des aéroports.
Une escale a une heure d'arrivée et une heure de départ.

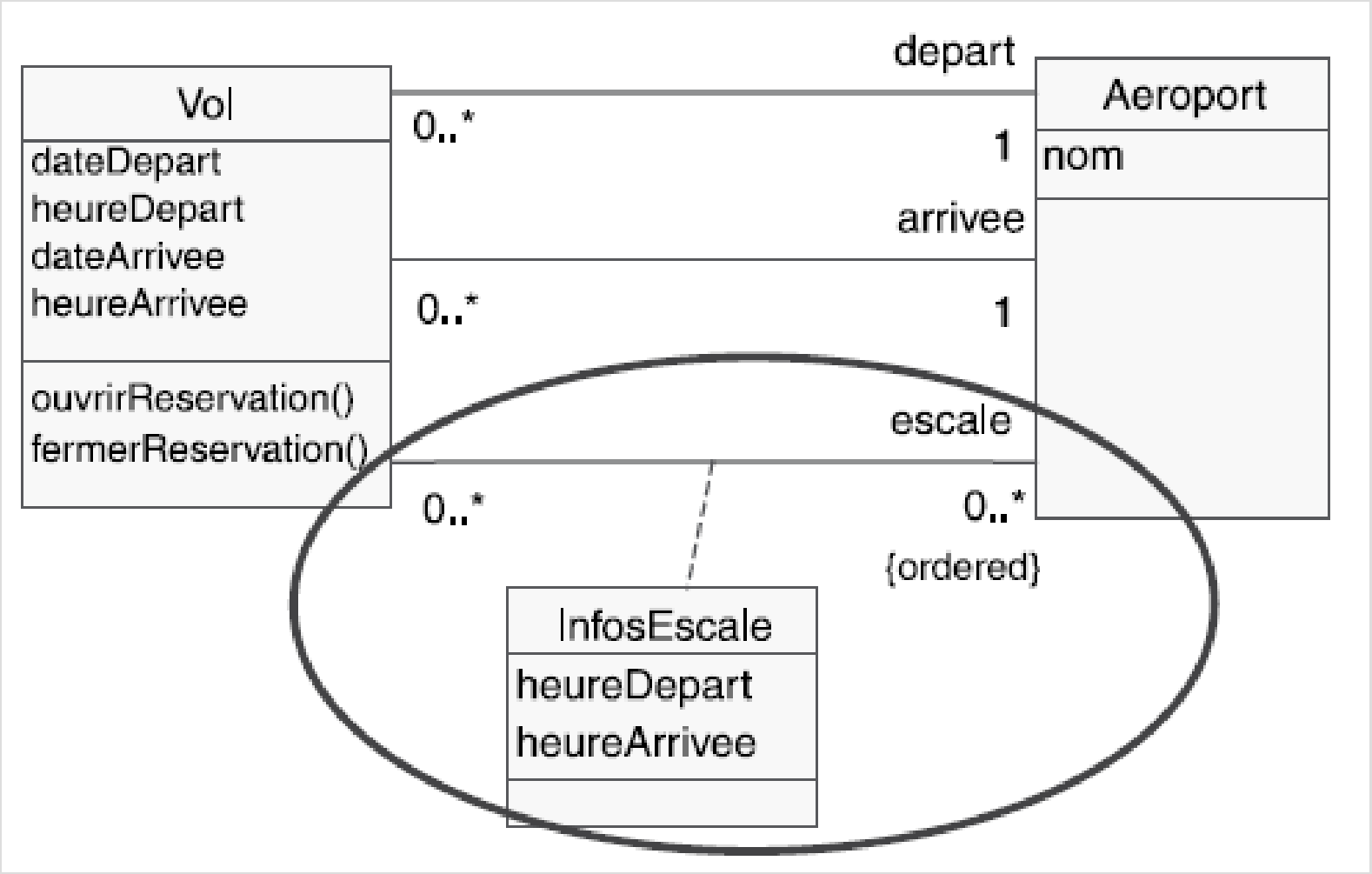


Vers quelque chose de plus élégant...



Un peu stupide...

Vers quelque chose de plus élégant : une classe d'association

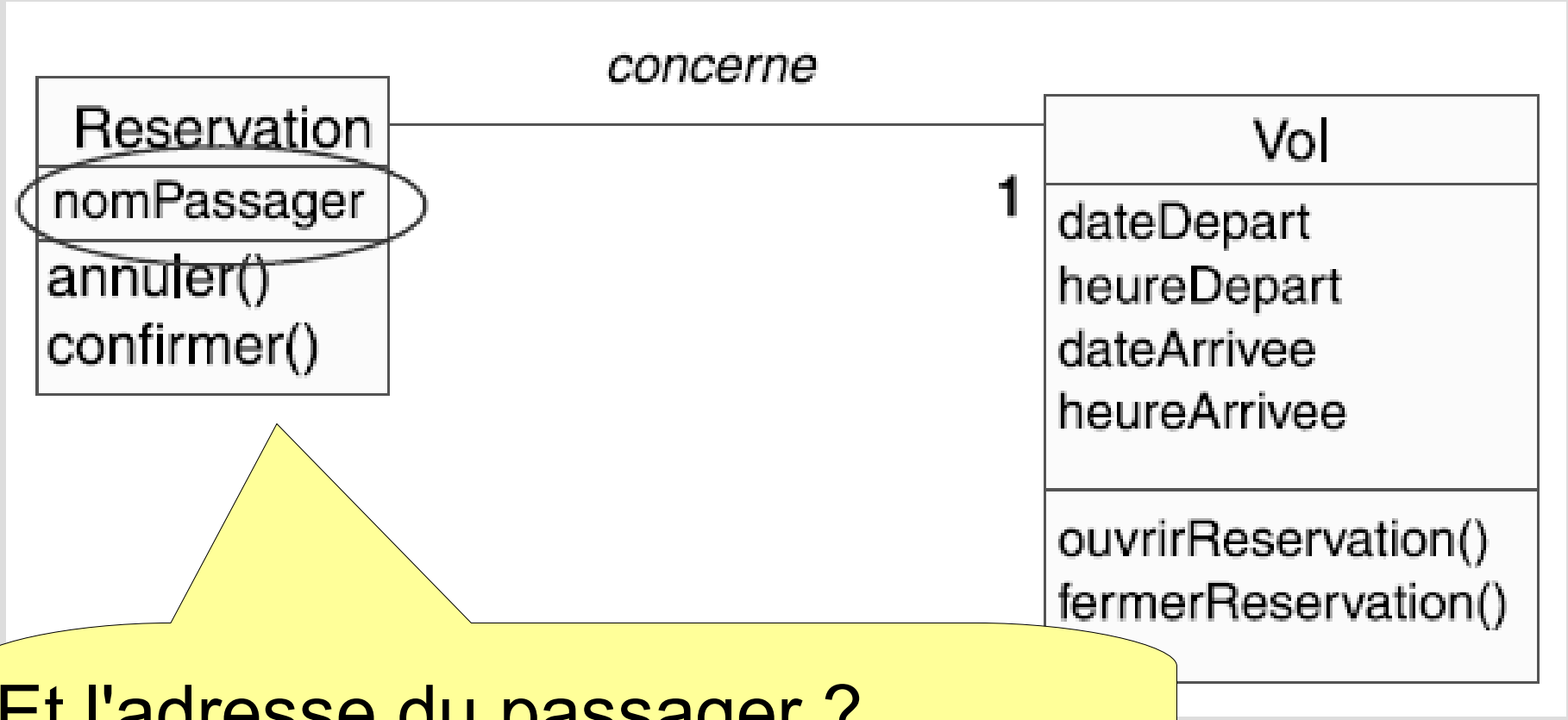


Une réservation concerne un seul vol, et un seul passager.
Une réservation peut être annulée ou confirmée.

Attributs !

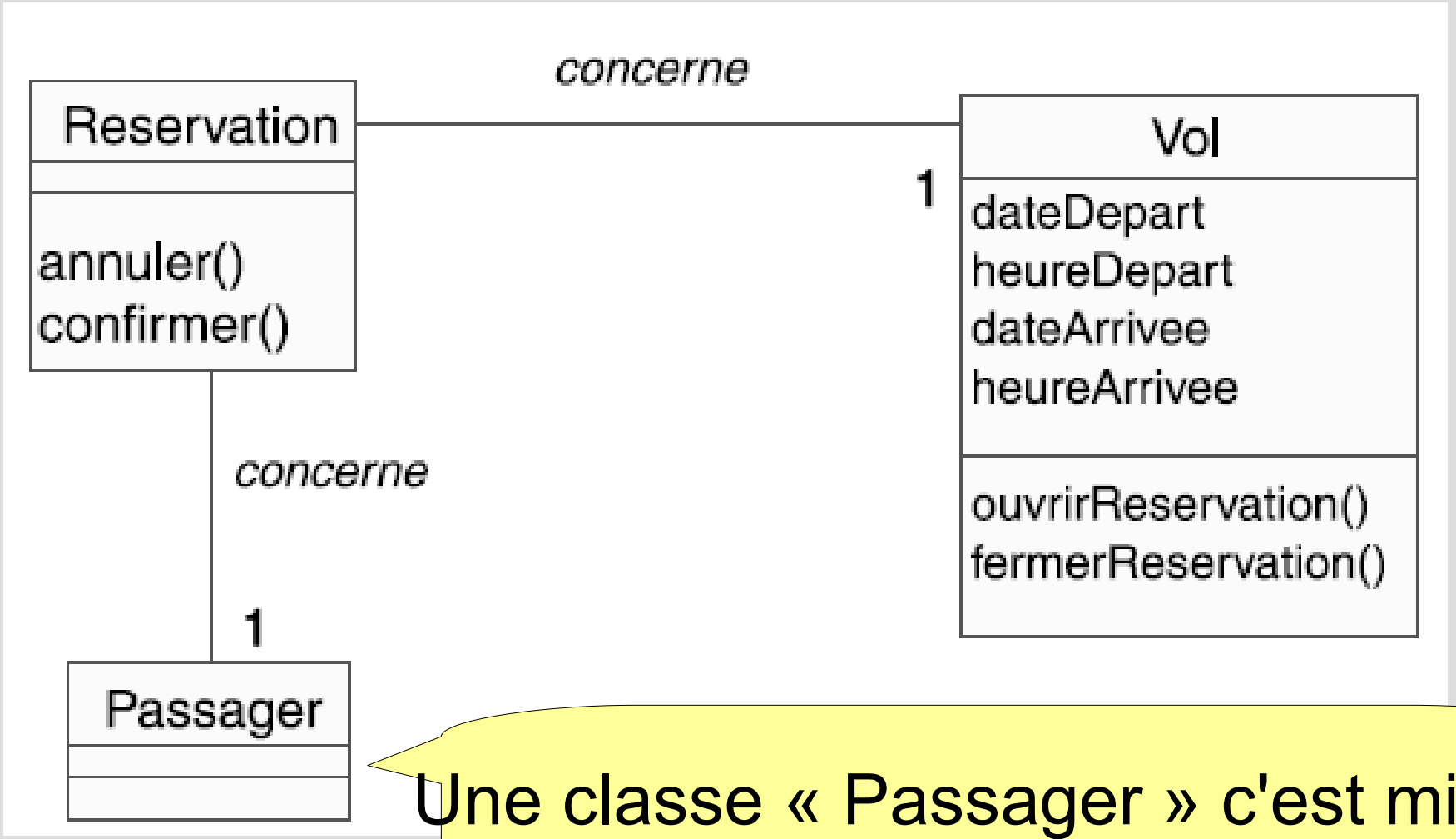
On crée une classe « Passager » ?

Une réservation concerne un seul vol, et un seul passager.
Une réservation peut être annulée ou confirmée.

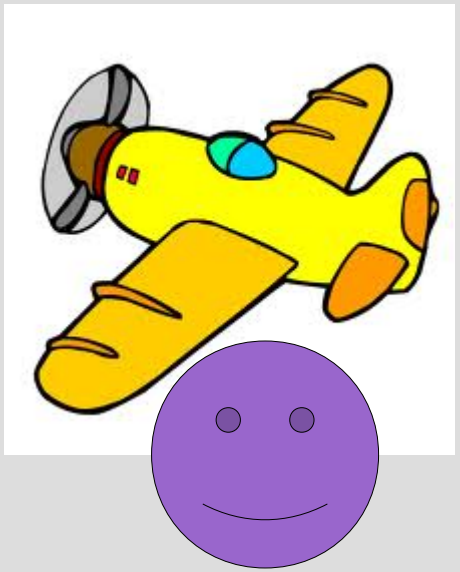


Et l'adresse du passager ?
Et les points fidélités du passager ?

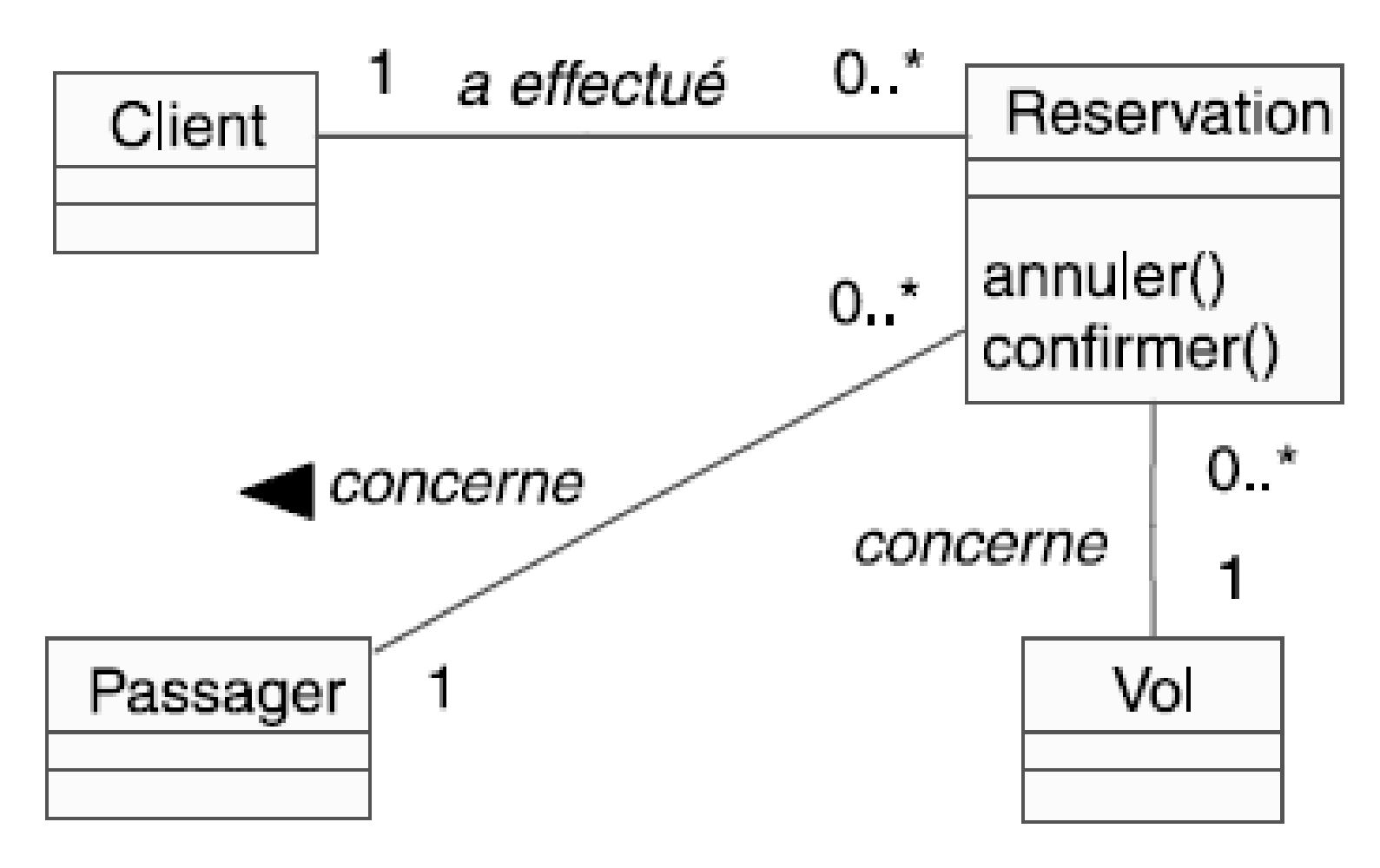
Une réservation concerne un seul vol, et un seul passager.
Une réservation peut être annulée ou confirmée.



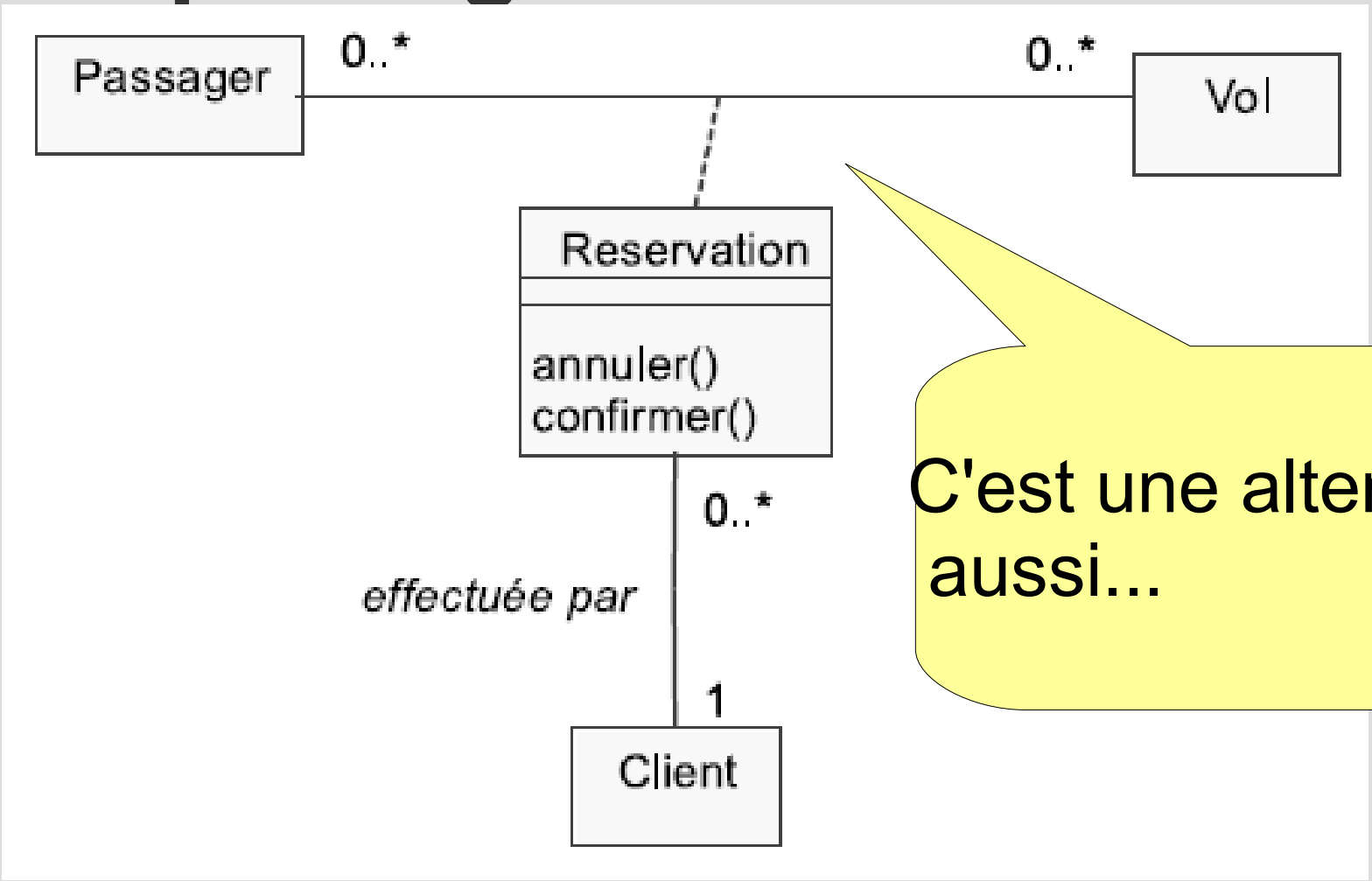
Un client peut réserver un ou plusieurs vols, pour des passagers différents.



Un client peut réserver un ou plusieurs vols, pour des passagers différents.

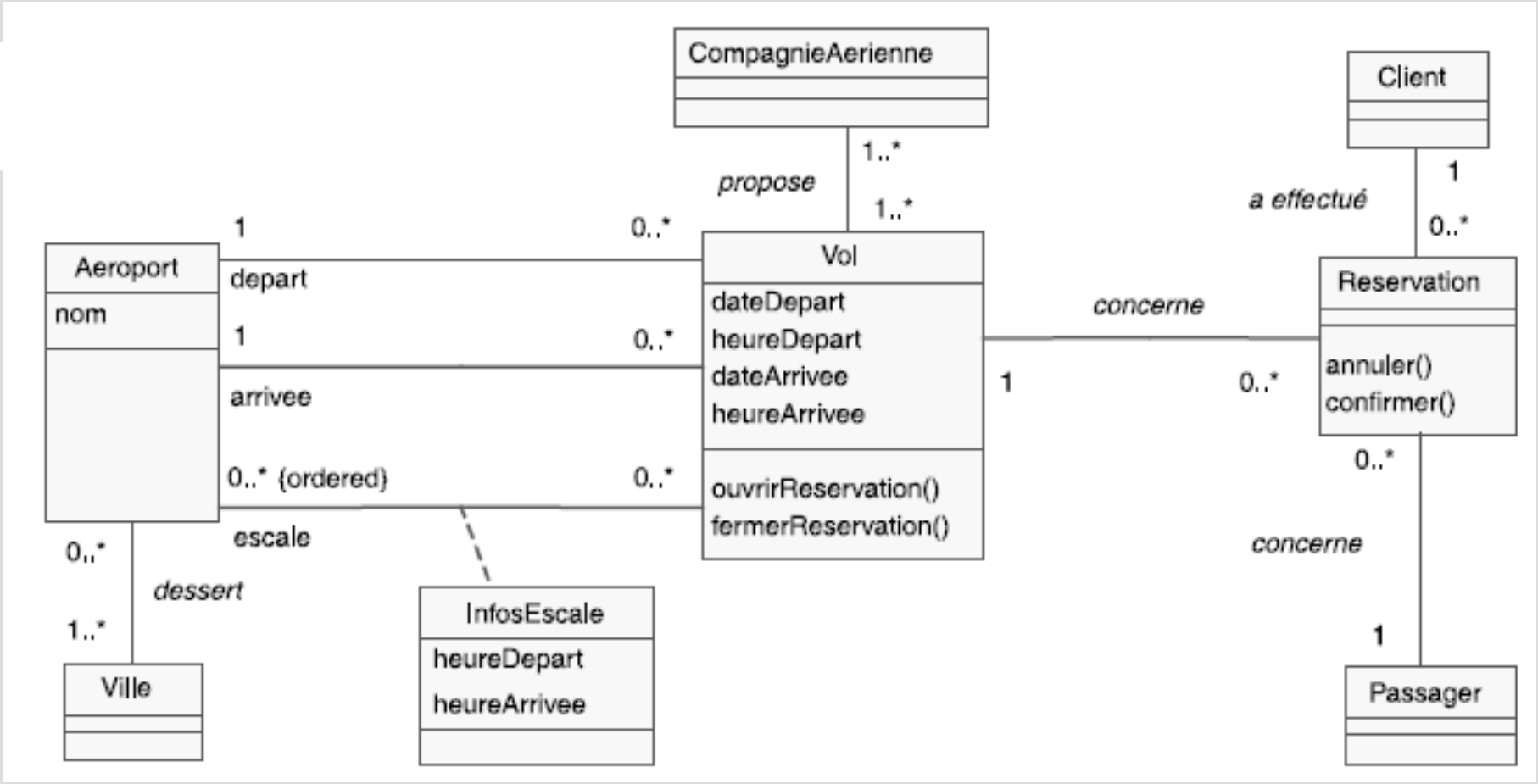


Un client peut réserver un ou plusieurs vols, pour des passagers différents.

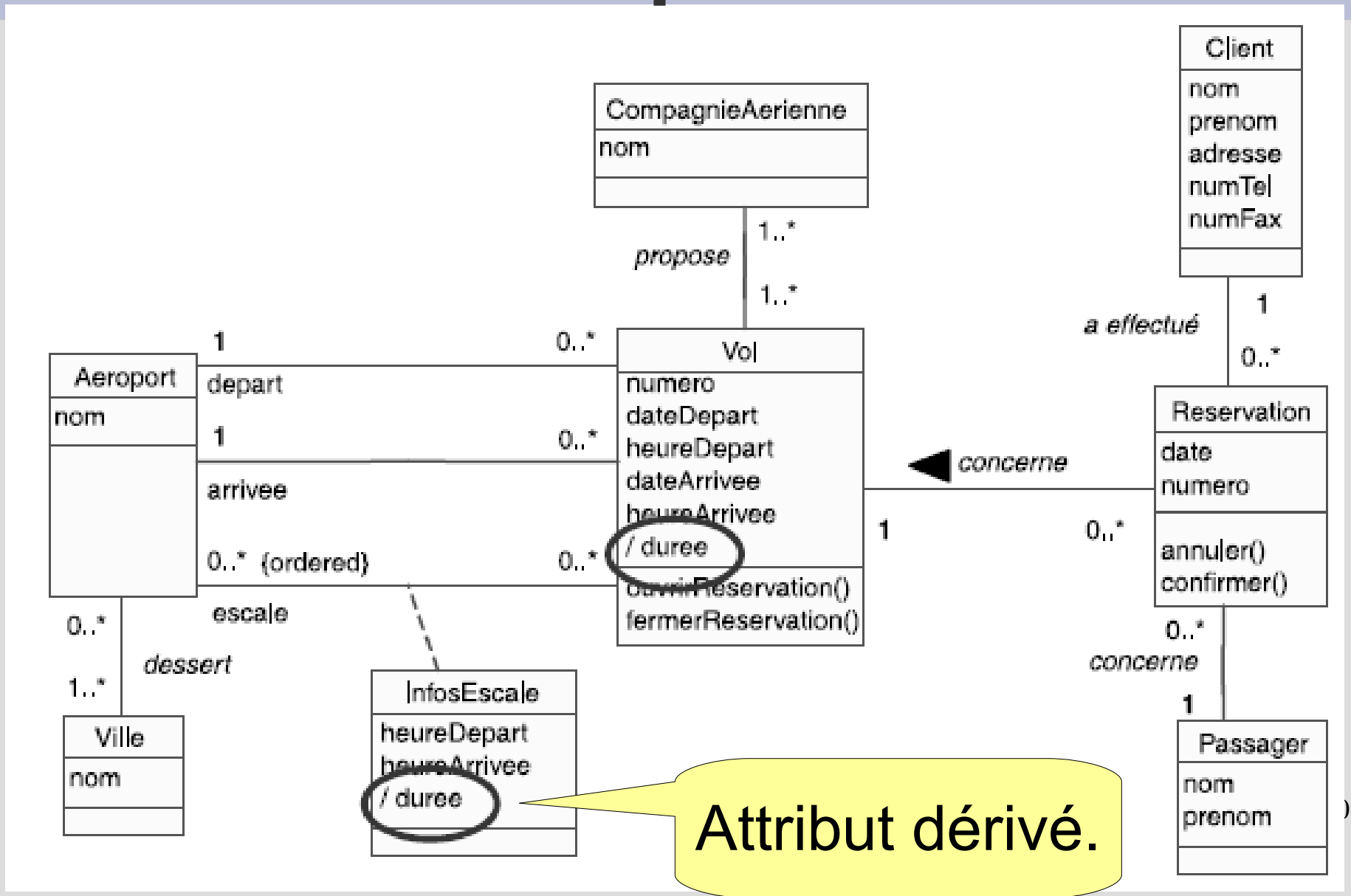


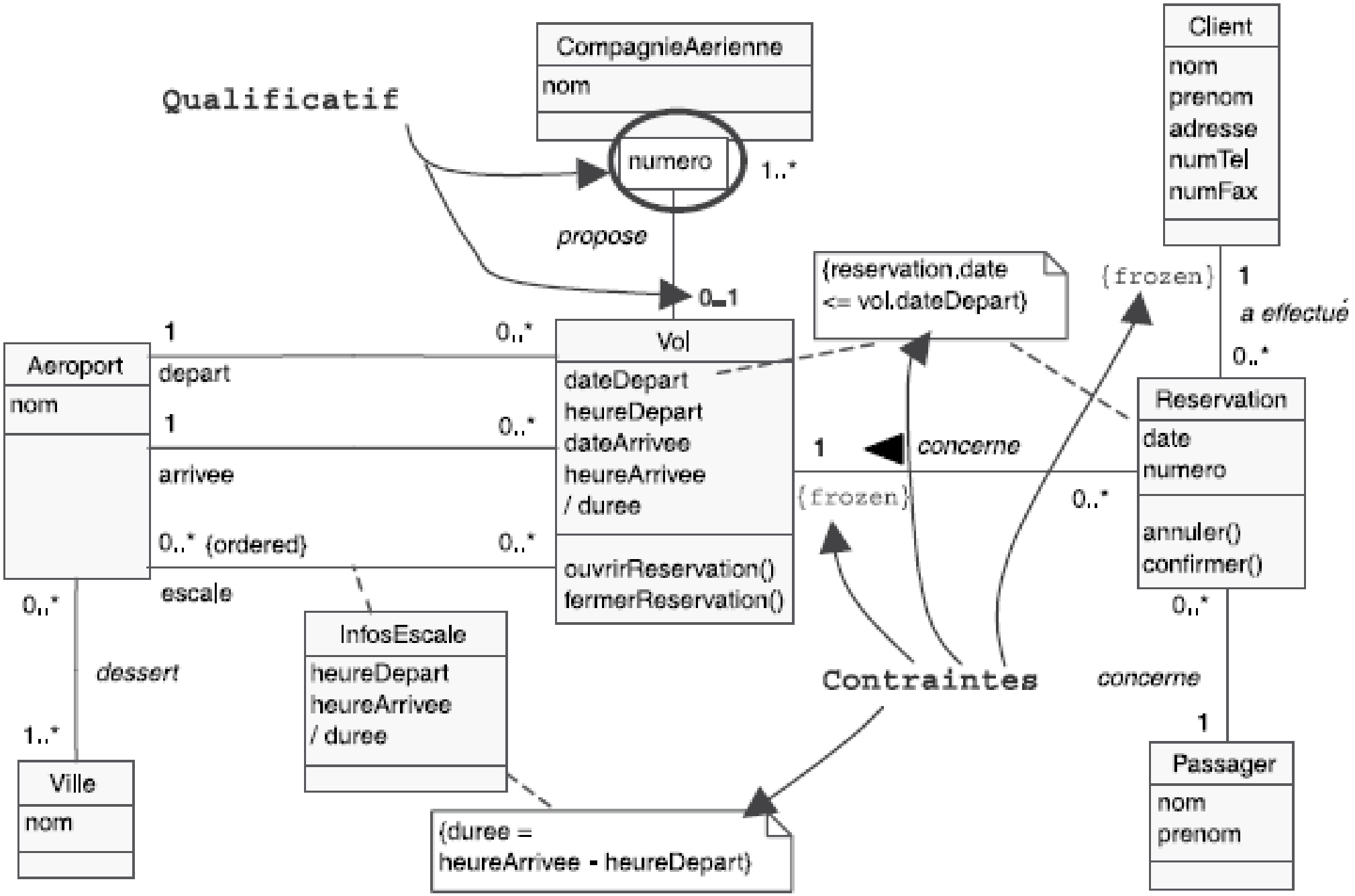
Quelques fioritures

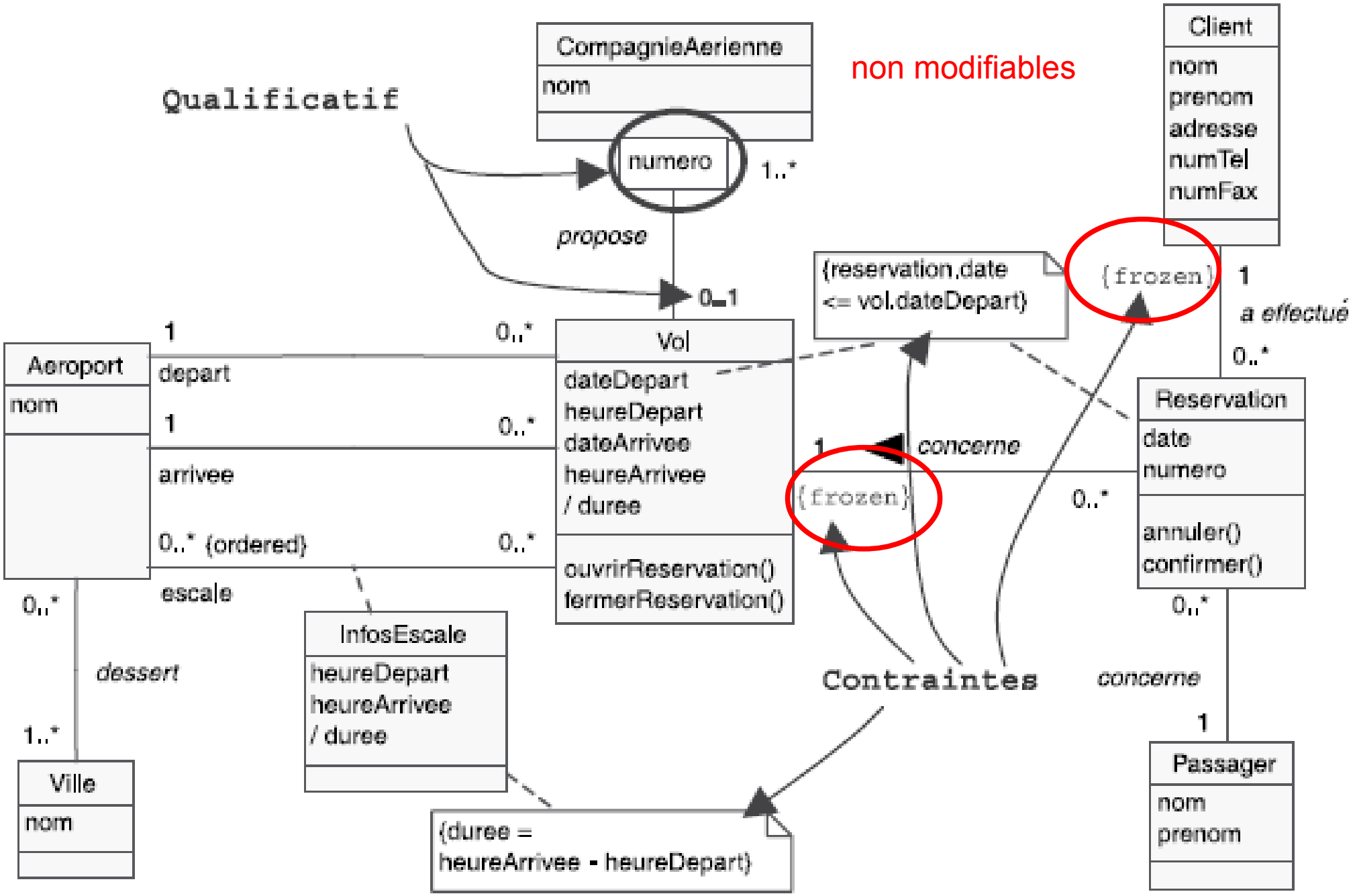
Le diagramme complet



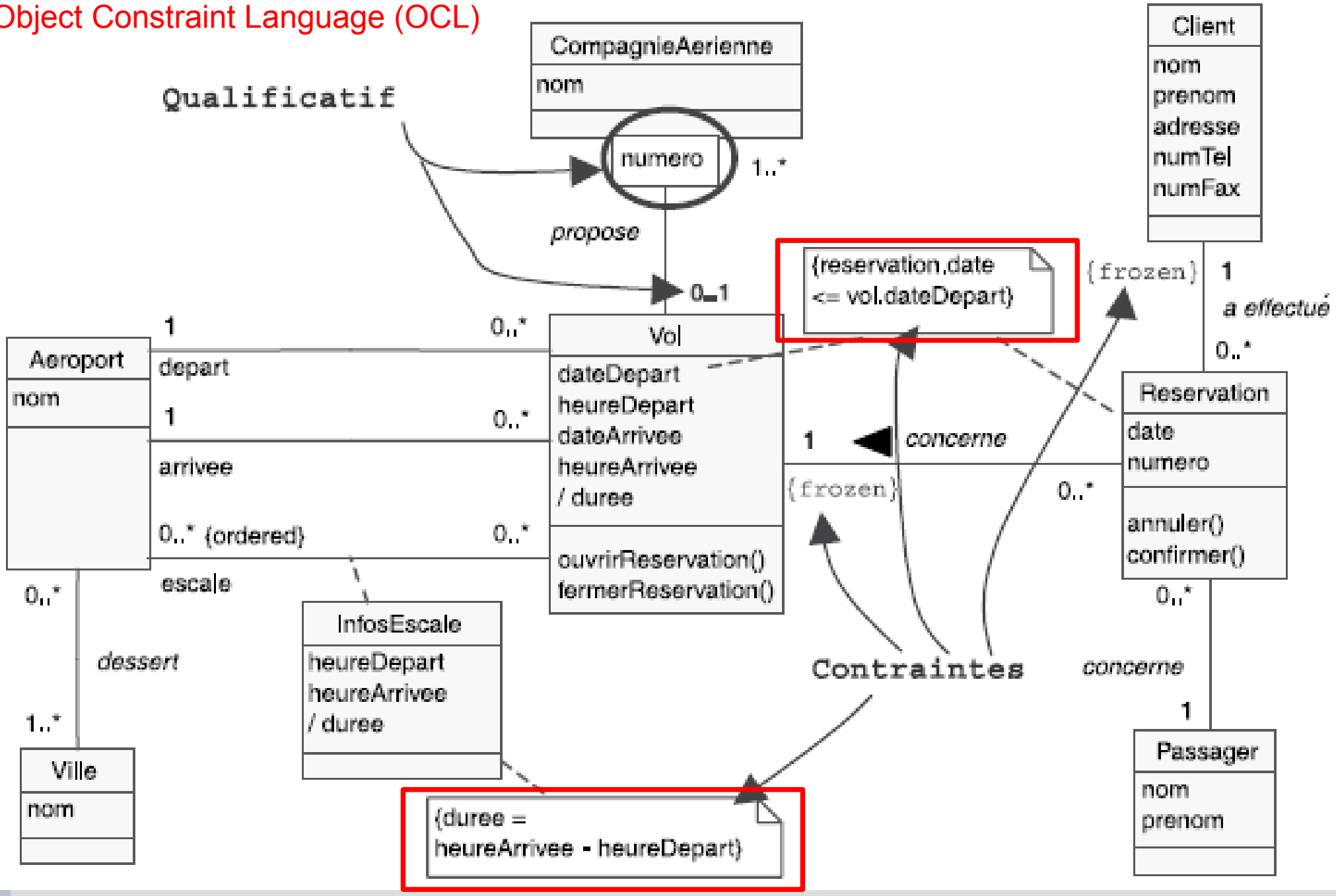
Le diagramme encore plus complet.




Qualificatif
CompagnieAerienne
nom
numero
1..*
propose
0..1
Client
nom
prenom
adresse
numTel
numFax
{reservation.date
<= vol.dateDepart}
{frozen}
1
a effectué
0..*
Aeroport
nom
1
depart
0..*
Vol
dateDepart
heureDepart
dateArrivee
heureArrivee
/ duree
ouvrirReservation()
fermerReservation()
1
0..*
arrivee
0..* {ordered}
0..*
escale
1
concerne
{frozen}
0..*
Reservation
date
numero
annuler()
confirmer()
0..*
0..*
dessert
1..*
Ville
nom
InfosEscale
heureDepart
heureArrivee
/ duree
Contraintes
concerne
1
Passager
nom
prenom
{duree =
heureArrivee - heureDepart}


Qualificatif
CompagnieAerienne
nom
numero
1..*
propose
0..1
non modifiables
Client
nom
prenom
adresse
numTel
numFax
{reservation.date
<= vol.dateDepart}
{frozen}
1
a effectué
0..*
Aeroport
nom
1
depart
0..*
Vol
dateDepart
heureDepart
dateArrivee
heureArrivee
/ duree
ouvrirReservation()
fermerReservation()
1
0..*
arrivee
0..* {ordered}
0..*
escale
1
concerne
{frozen}
0..*
Reservation
date
numero
annuler()
confirmer()
0..*
dessert
1..*
Ville
nom
InfosEscale
heureDepart
heureArrivee
/ duree
Contraintes
concerne
1
Passager
nom
prenom
{duree =
heureArrivee - heureDepart}

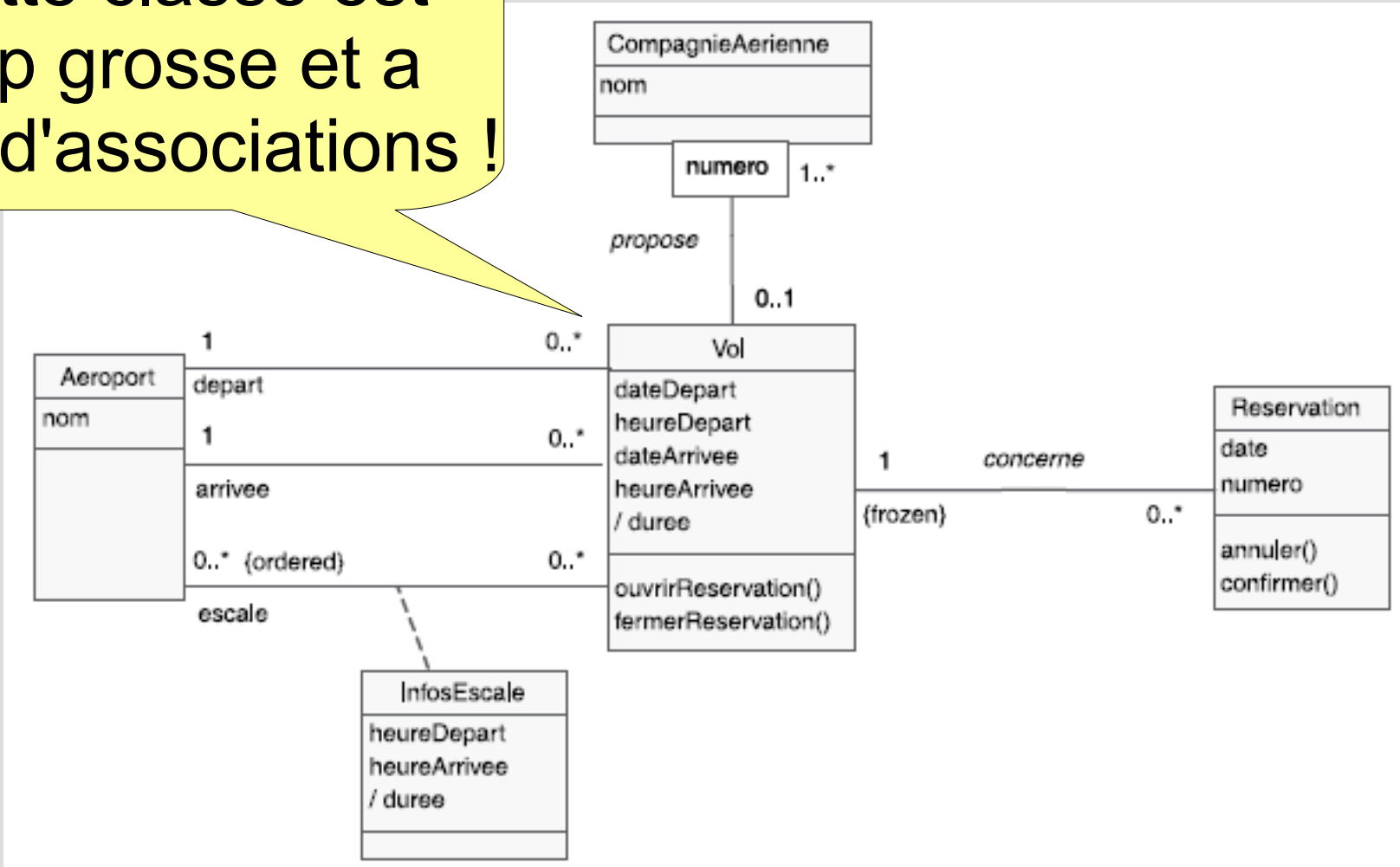
Object Constraint Language (OCL)



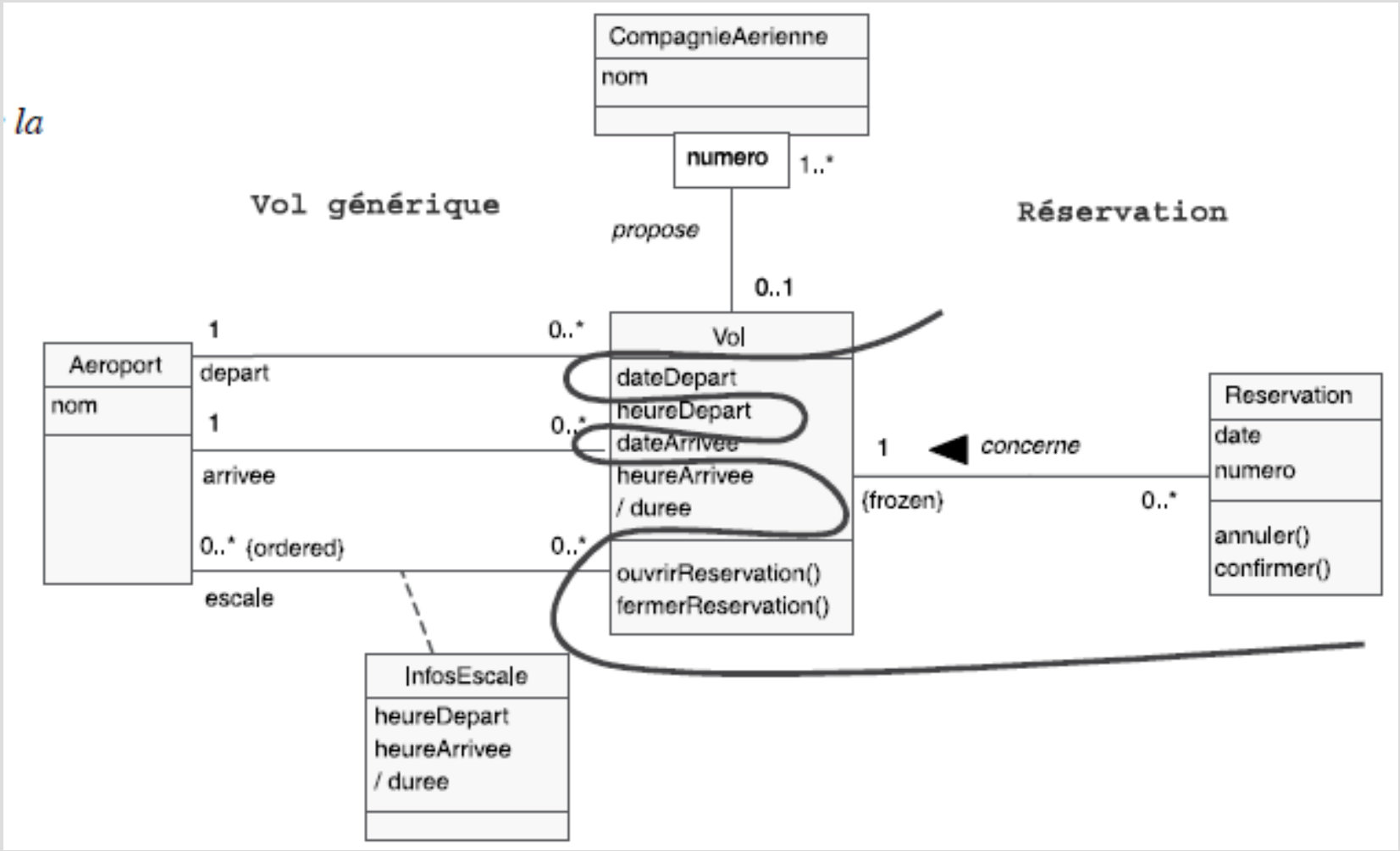
Forte cohésion

Forte cohésion de la classe « Vol »

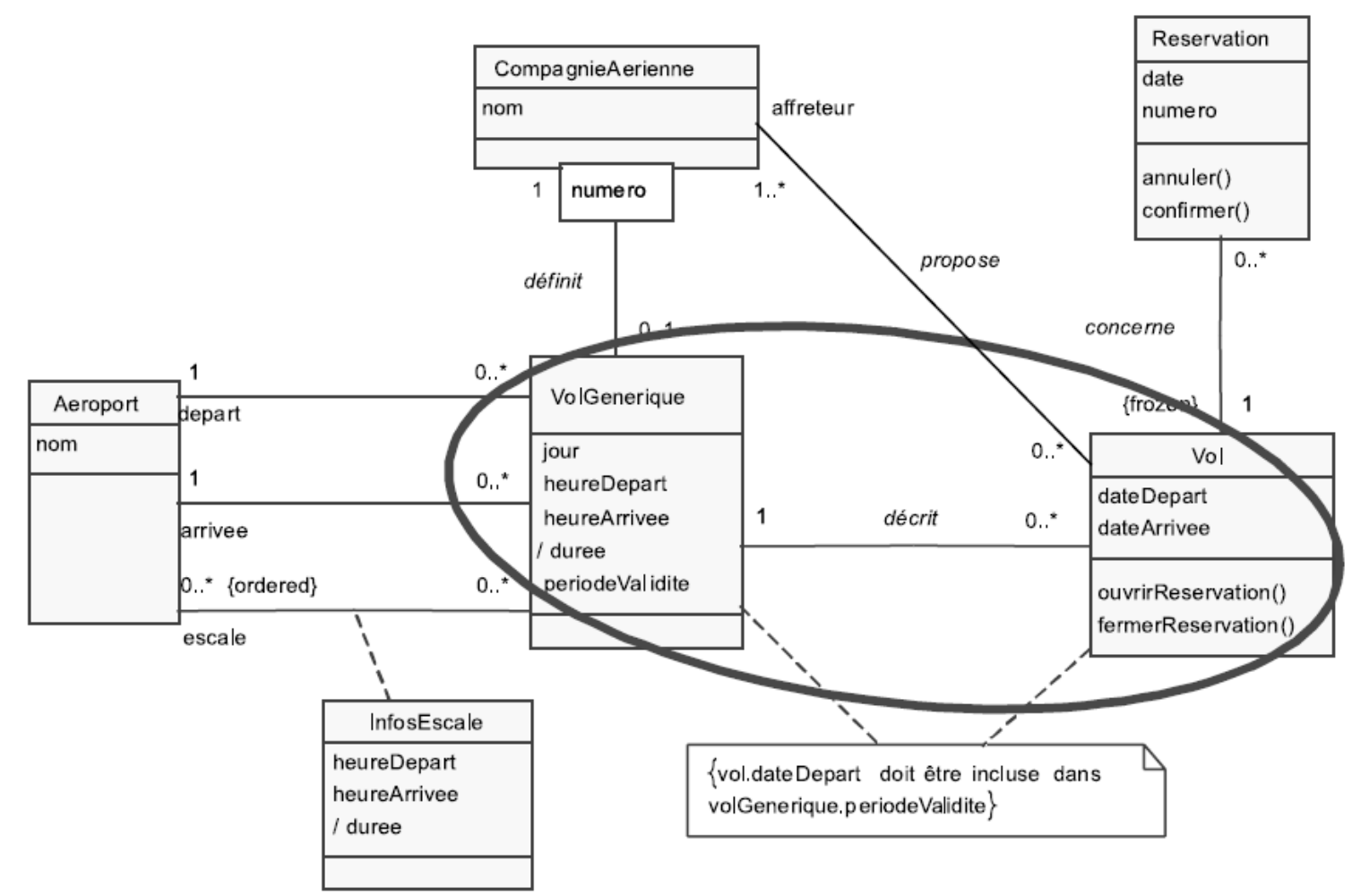
Cette classe est trop grosse et a trop d'associations !

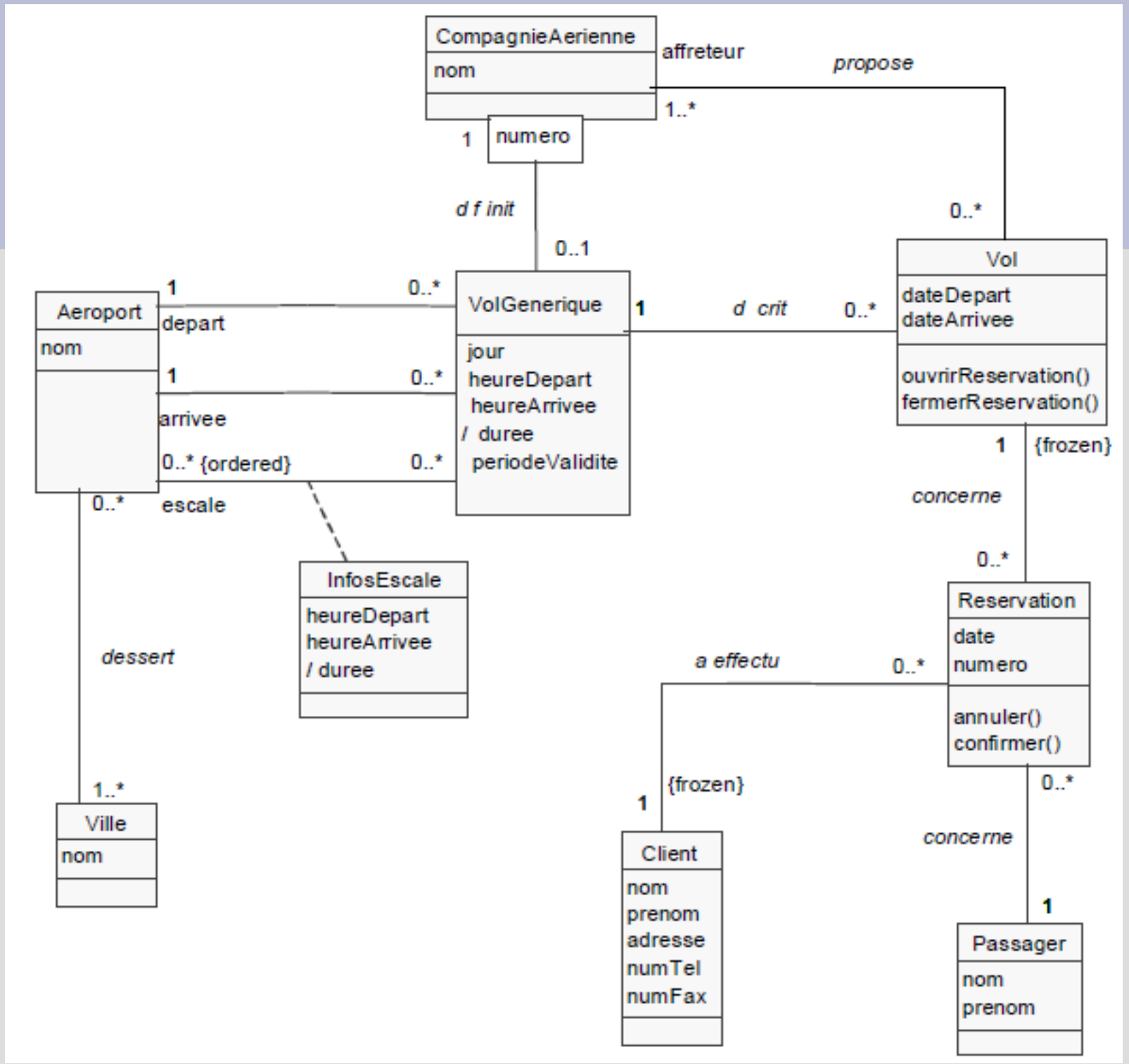


Les vols génériques !



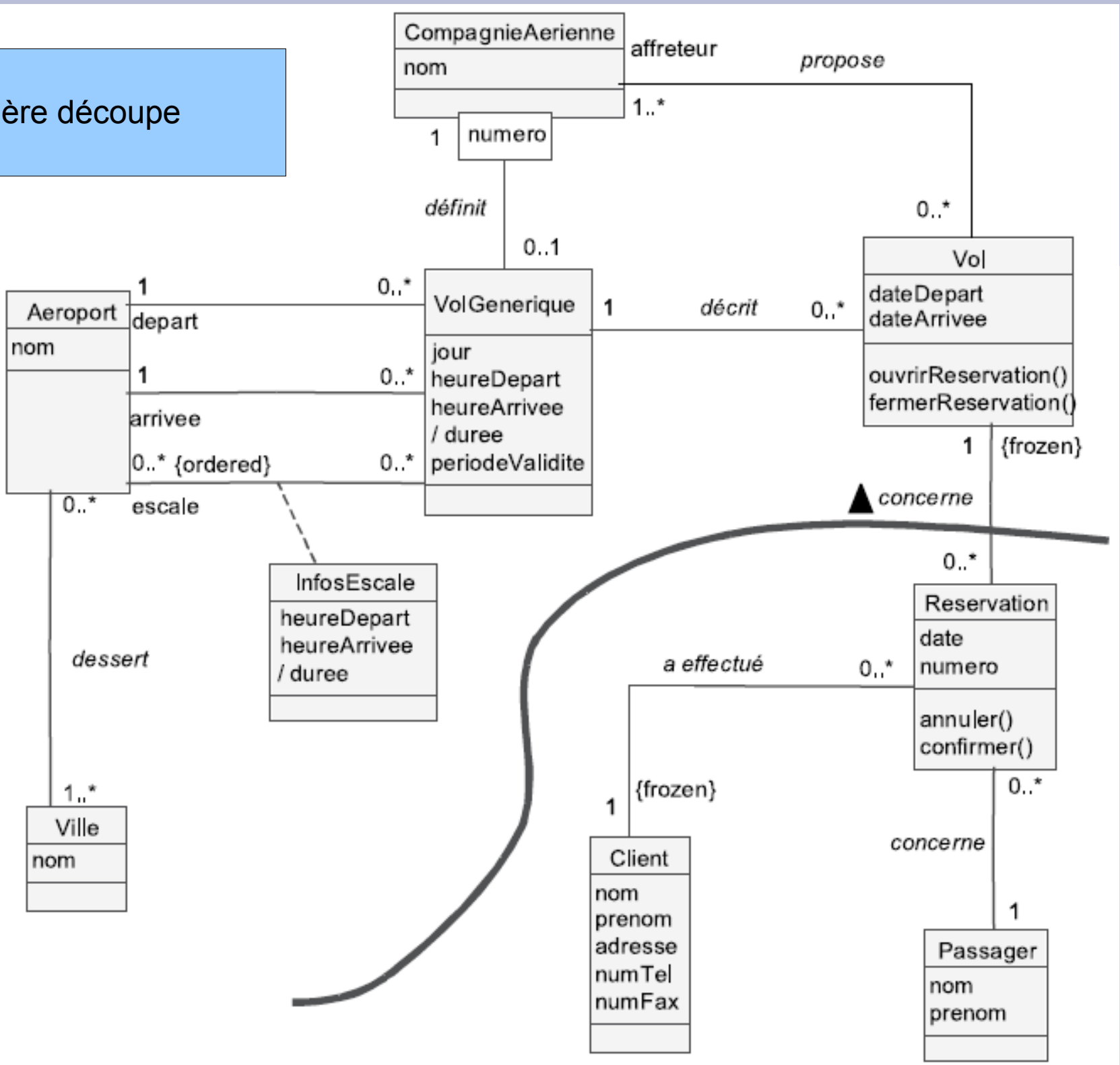
Les vols génériques !



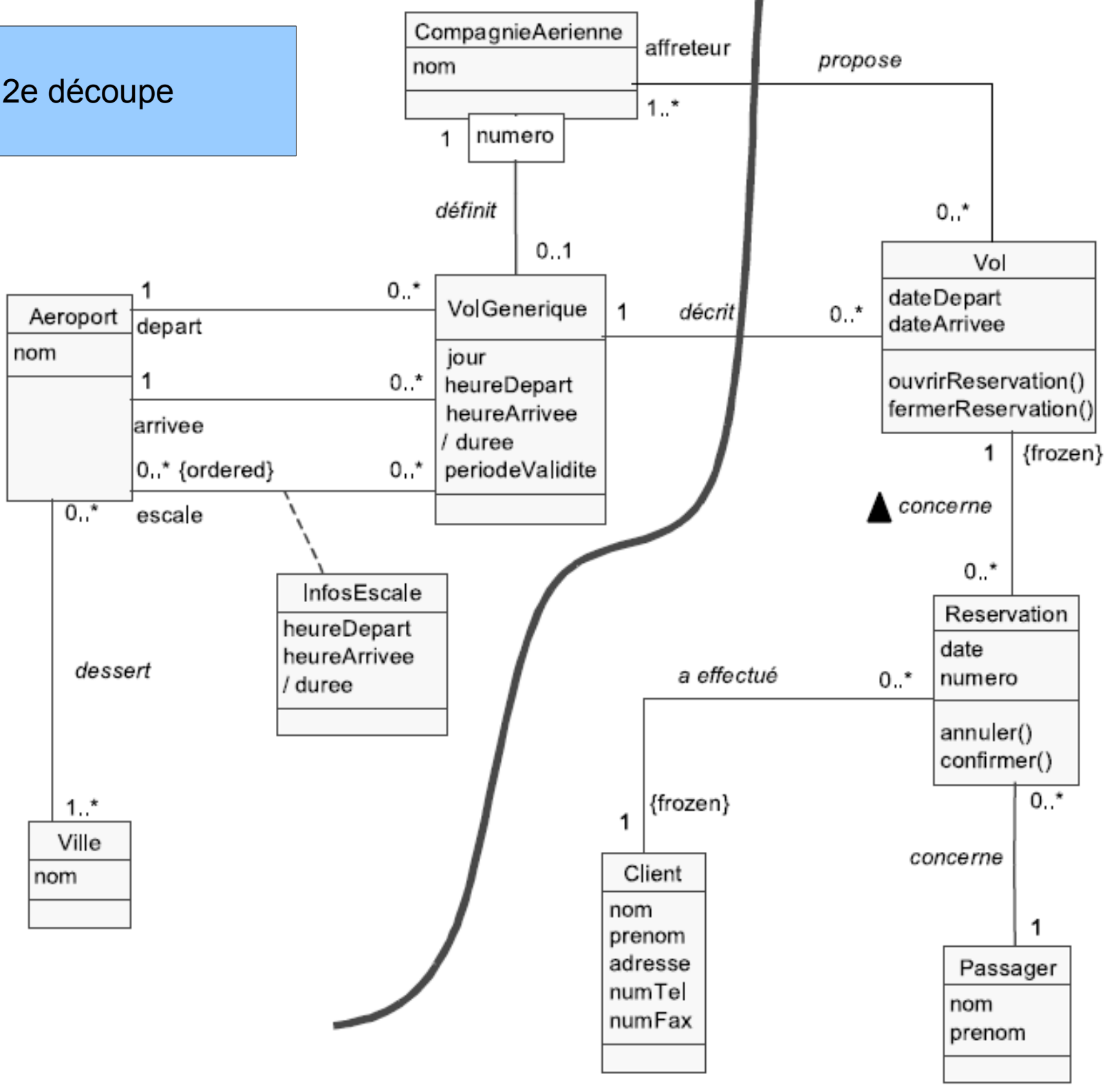

CompagnieAerienne
nom
affreteur
propose
1..*
1
numero
d f init
0..1
0..*
Vol
dateDepart
dateArrivee
ouvrirReservation()
fermerReservation()
Aeroport
nom
1
depart
0..*
VolGenerique
jour
heureDepart
heureArrivee
/ duree
periodeValidite
1
d crit
0..*
1
arrivee
0..*
0..*
{ordered}
0..*
escale
0..*
1
{frozen}
concerne
0..*
InfosEscale
heureDepart
heureArrivee
/ duree
Reservation
date
numero
annuler()
confirmer()
dessert
a effectu
0..*
1..*
Ville
nom
{frozen}
1
0..*
concerne
Client
nom
prenom
adresse
numTel
numFax
1
Passager
nom
prenom

Découpe en packages

1ère découpe

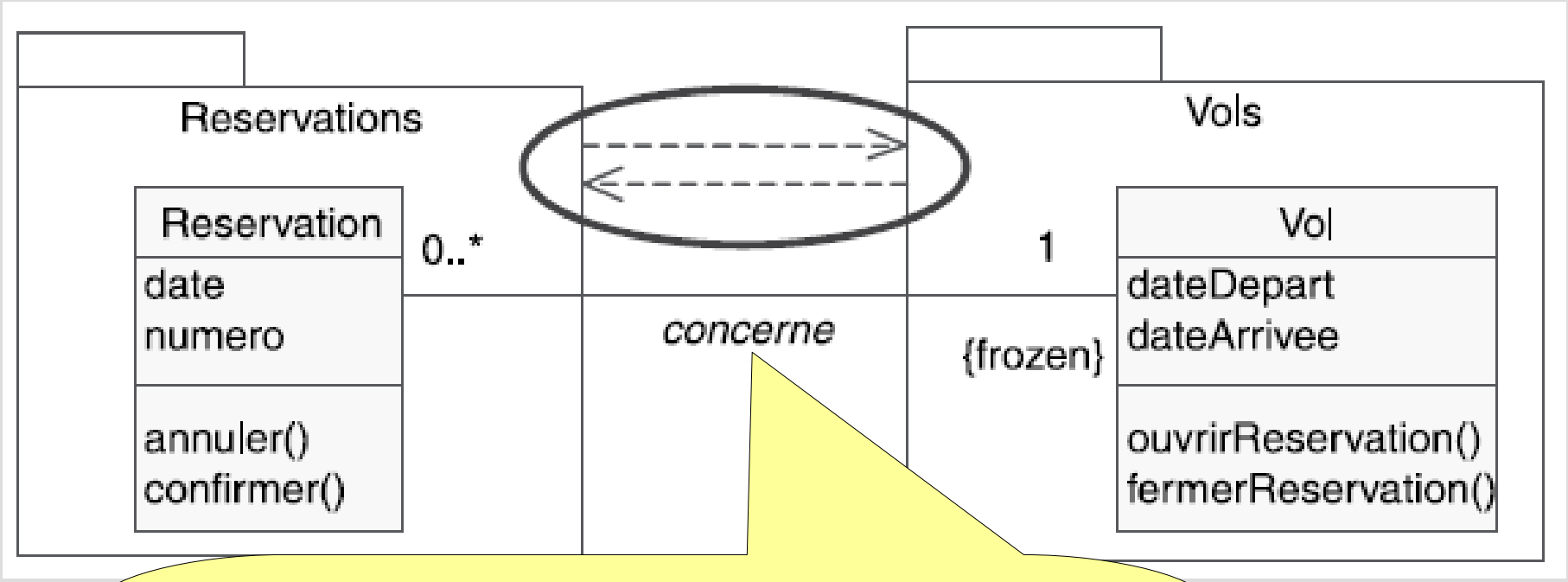


2e découpe


CompagnieAerienne
nom
affreteur
propose
1..*
1
numero
définit
0..1
0..*
Vol
dateDepart
dateArrivee
ouvrirReservation()
fermerReservation()
Aeroport
nom
1
depart
0..*
VolGenerique
1
décrit
0..*
jour
heureDepart
heureArrivee
/ duree
periodeValidite
1
arrivee
0..*
0..* {ordered}
0..*
0..*
escale
1
{frozen}
concerne
0..*
InfosEscale
heureDepart
heureArrivee
/ duree
Reservation
date
numero
annuler()
confirmer()
dessert
a effectué
0..*
1..*
Ville
nom
1
{frozen}
0..*
concerne
Client
nom
prenom
adresse
numTel
numFax
1
Passager
nom
prenom

Dépendance mutuelle

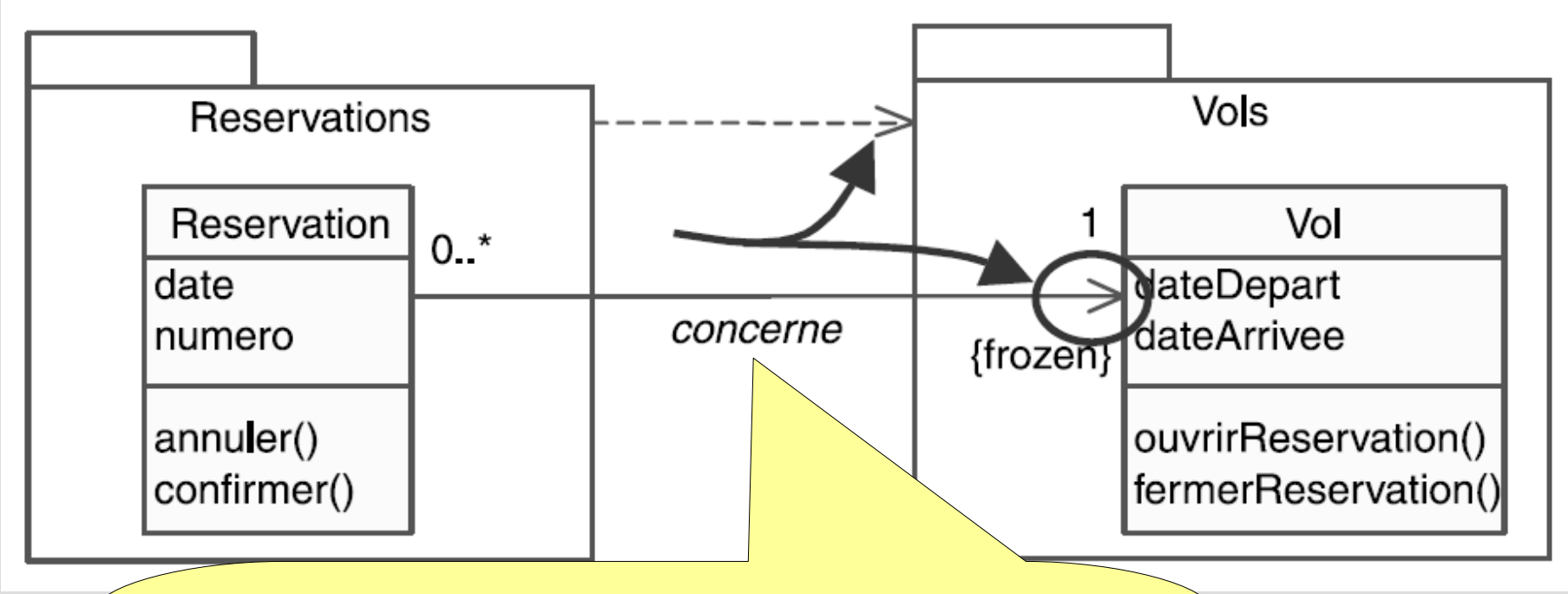
1ère découpe



Pas de modularité
Application difficile à maintenir !

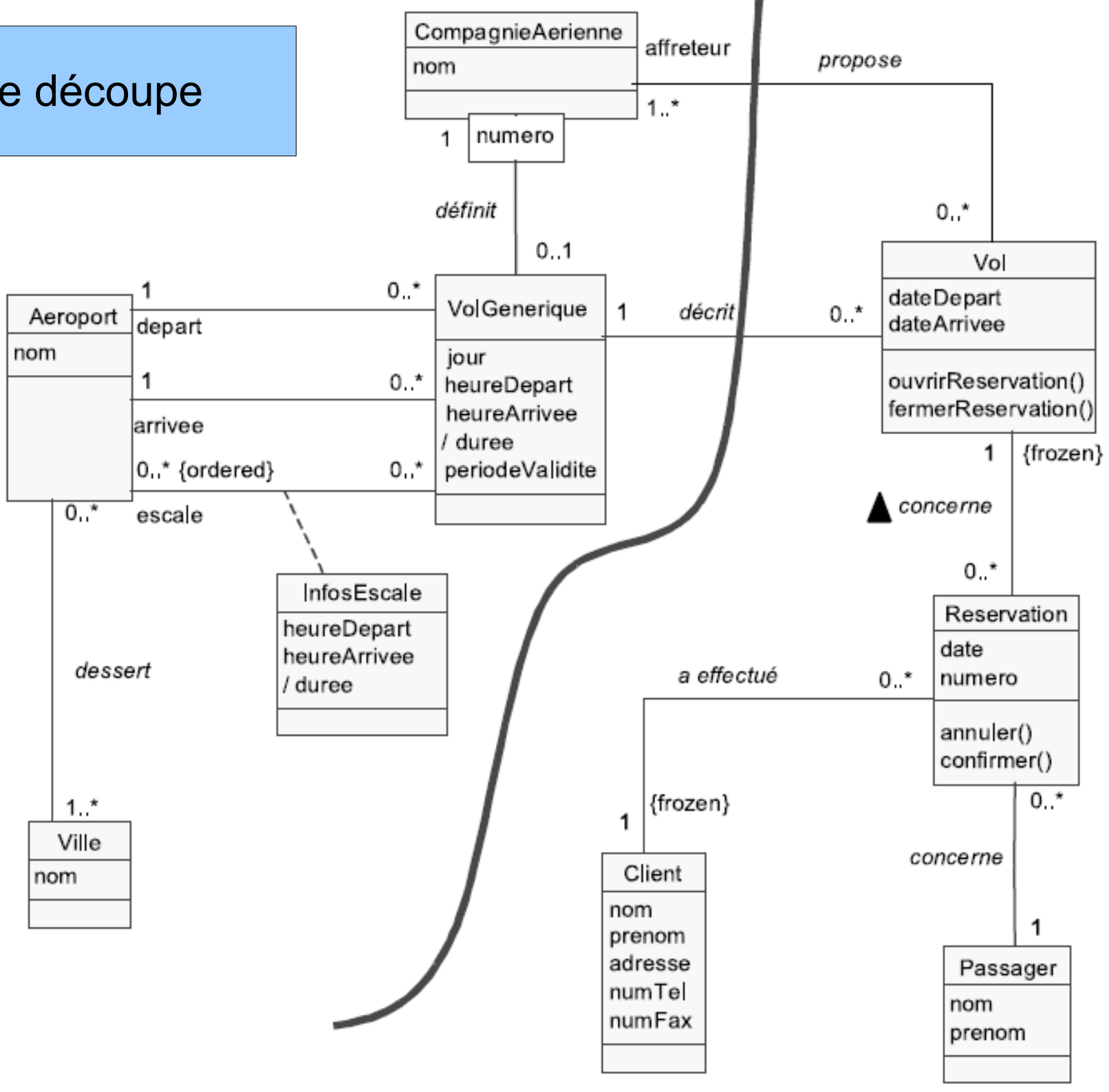
Solution : on privilégie un sens de navigation

1ère découpe



Un vol n'a pas besoin de connaître les réservations associées !

2e découpe


CompagnieAerienne
nom
affreteur
propose
1..*
1
numero
définit
0..1
0..*
Vol
dateDepart
dateArrivee
ouvrirReservation()
fermerReservation()
1
{frozen}
Aeroport
nom
1
depart
0..*
1
arrivee
0..*
0..* {ordered}
0..*
escale
0..*
VolGenerique
jour
heureDepart
heureArrivee
/ duree
periodeValidite
1
décrit
0..*
concerne
0..*
InfosEscale
heureDepart
heureArrivee
/ duree
Reservation
date
numero
annuler()
confirmer()
a effectué
0..*
dessert
1..*
Ville
nom
1
{frozen}
0..*
concerne
Client
nom
prenom
adresse
numTel
numFax
1
Passager
nom
prenom

Dépendance mutuelle incassable !

2e découpe

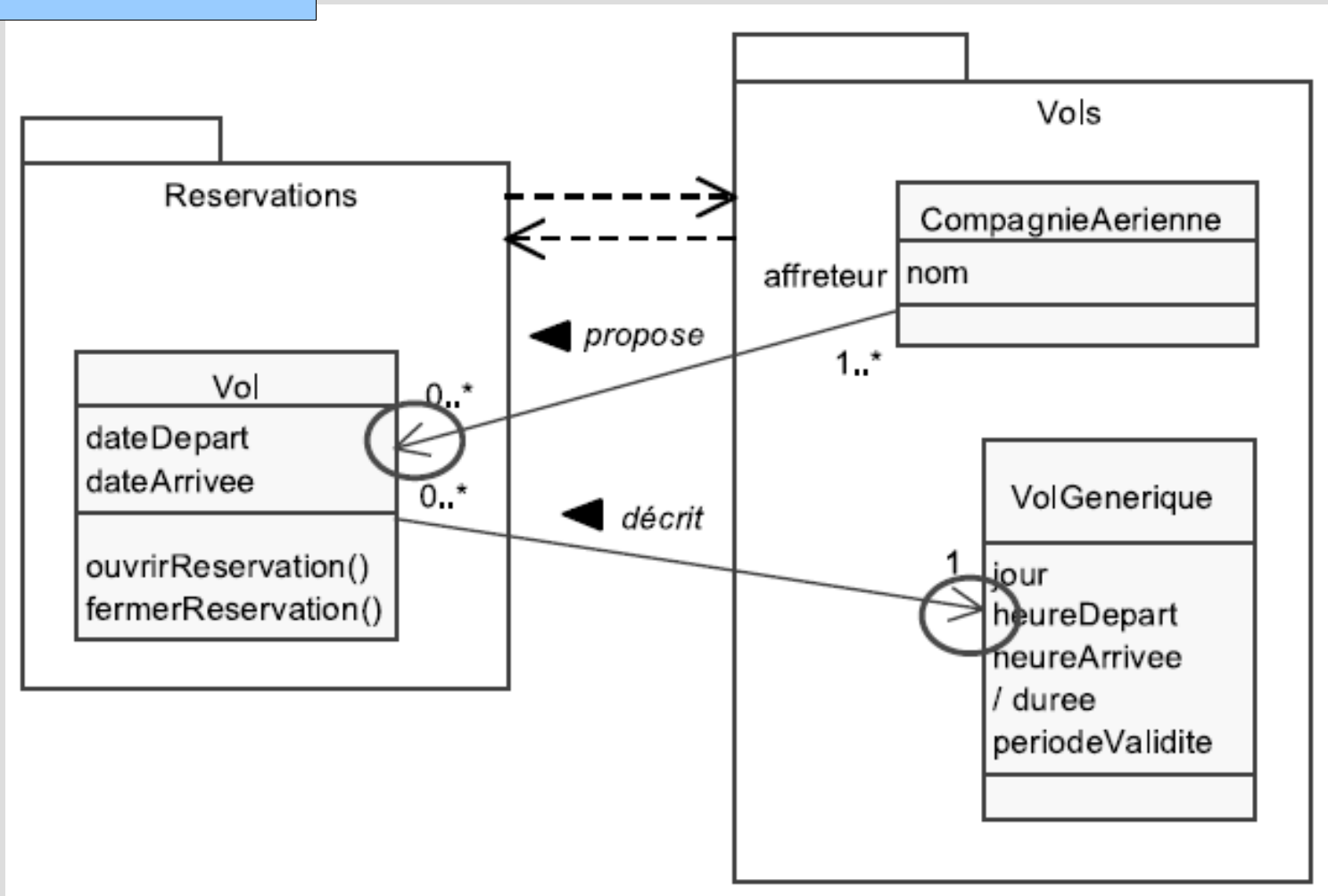
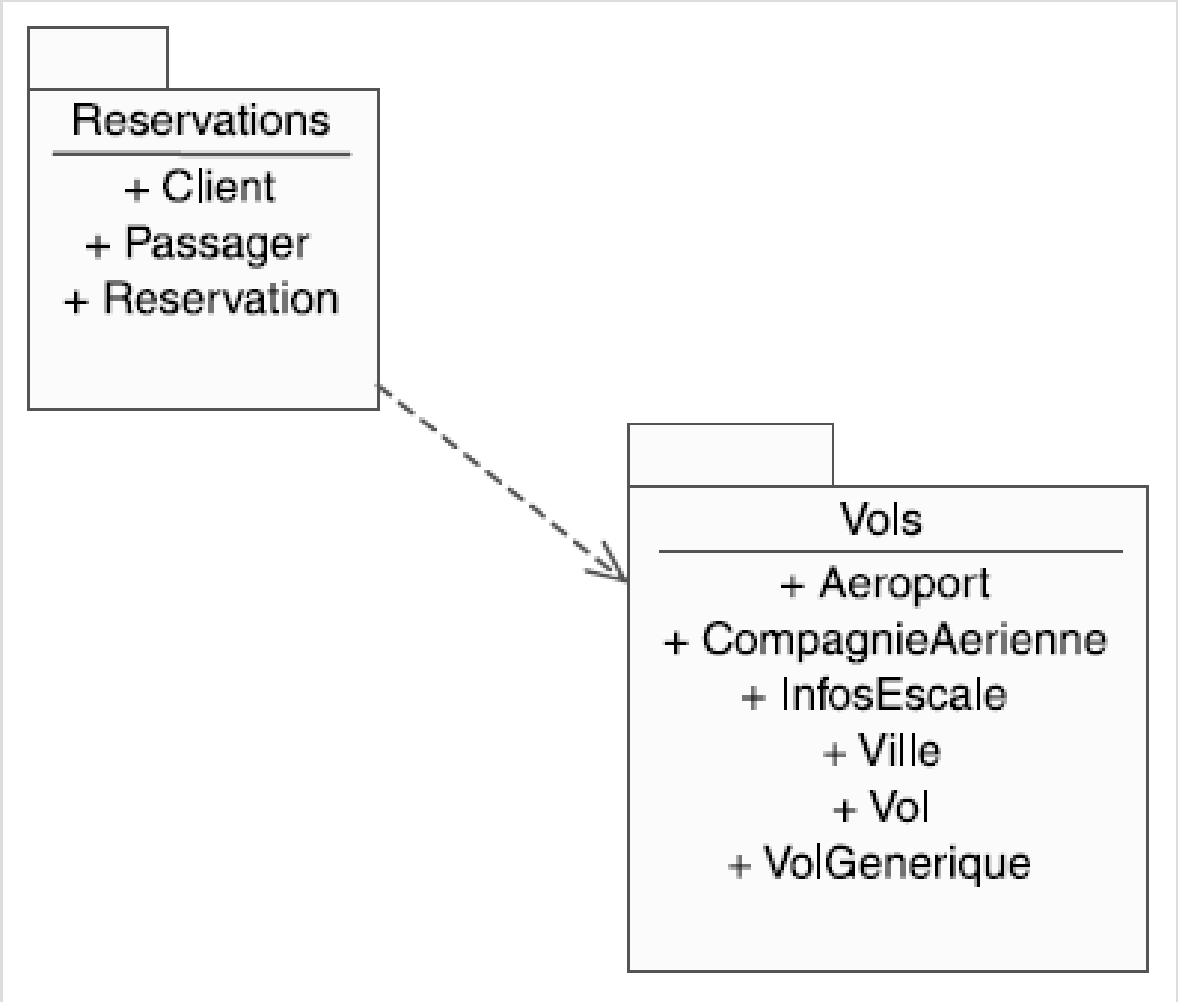


Diagramme de package de la solution retenue

1ère découpe



Inverser les dépendances