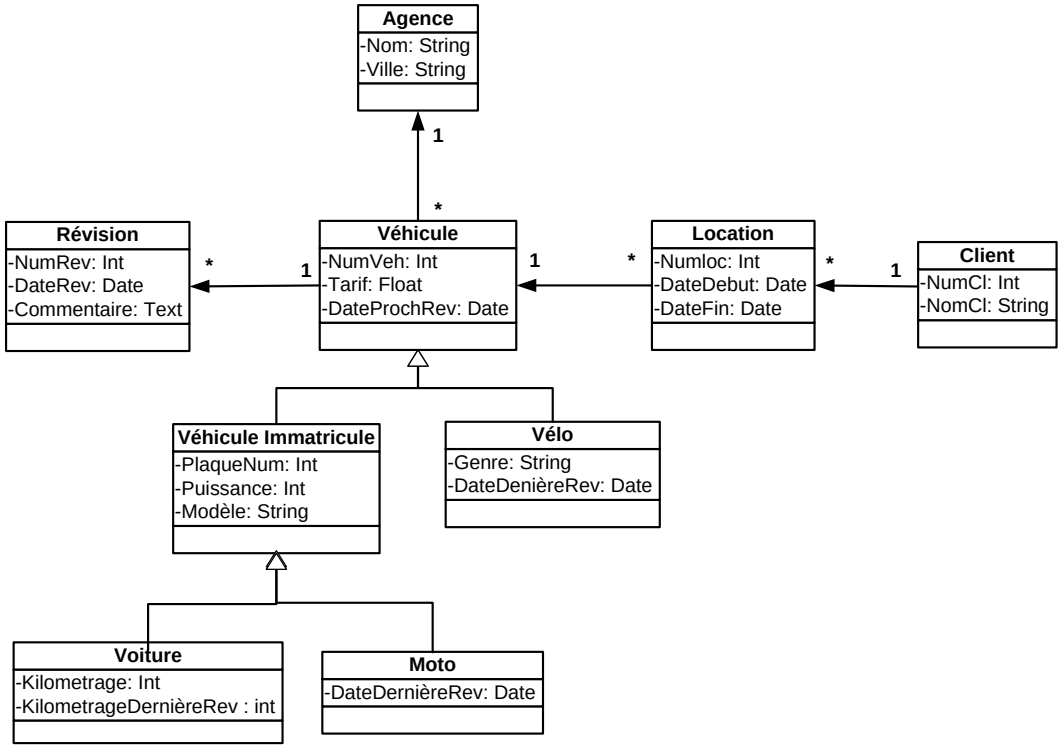
Corrigé de l'exercice 03 :

On considère le diagramme de classes suivant, décrivant une base de données modélisant un système de location de véhicules.



On veut modéliser ce schéma en SQL3. On décrit les associations dans le sens des flèches (ex : Véhicule→ Agence, c-à-d qu'on ne sauvegarde dans la BD que le fait qu'un véhicule est géré par une agence).

Questions :

1. Définir les types **T_Agence**, **T_Revision, T_Vehicule**, **T_VehiculeImmatricule** et **T_Voiture**.

```
create type t_agence as object(nom varchar2(100), ville varchar2(50));
/
create type t_revision as object(numrev integer, daterev date, commentaire varchar2(200));
/
create type t_set_ref_revision as table of ref t_revision;
/
create type t_vehicule as object(numveh integer, tarif float, dateprochrev date, gere_par ref t_agence, revision t_set_ref_revision ) not final;
/
create type t_vehicule_immatricule under t_vehicule(plaquenum integer, puissance integer, modele varchar2(50)) not final;
/
create type t_velo under t_vehicule(genre varchar2(50), datedernierерev date);
/
create type t_voiture under t_vehicule_immatricule(kilometrage integer);
/
create type t_moto under t_vehicule_immatricule(datedernierерev date);
/
```

2. Définir les types **T_Location** et **T_Client**.

```
create or replace type t_location as object (numloc integer, datedebut date, datefin date, vehicule ref t_vehicule);
/
create type t_set_ref_location as table of ref t_location;
/
create type t_client as object (numero integer, nom varchar2(50), locations t_set_ref_location);
/
```

3. Définir les tables **Agences** stockant les objets de type T_Agences, **Vehicules** stockant les objets

de type T_Vehicule, **Locations** stockant les objets de type T_Location et **Clients** stockant les objets de type T_Client.

```
create table agences of t_agence(primary key(nom));
create table clients of t_client(primary key (numero))
nested table locations store as table_locations;
create table vehicules of t_vehicule(primary key(numveh))
nested table revision store as table_revisions;
create table locations of t_location(primary key(numloc));
```

4. La table Agences contient l'agence « Azimut Car à Bordj El Kiffan » ; Ecrire l'instruction SQL3 permettant d'insérer la voiture de numéro 111, tarif de location est 3500 DA, immatriculée 1245610816, de modèle PEUGEOT NEW 208, de puissance 136 chevaux et son kilométrage est à 20000 KM pour l'agence « Azimut Car ».

```
insert into vehicules values(t_voiture('111','3500','',(select ref(a) from agences a where a.nom='Azimut Car'),
t_set_ref_revision(),'1245610816','136','PEUGEOT  NEW 208', '20000'));
```

5. La table Clients contient le client « 102, Ahmed MESSAOUDI ». Ce client a loué la voiture N° 111 pour la période de 04/09/2019 à 09/09/2019. Ecrire l'instruction SQL3 permettant d'ajouter cette location de N° 1058 pour ce client.

```
//Ajouter la location
insert into locations values('1058', '04/09/2019', '09/09/2019',(select ref(v) from vehicules v
where v.numveh='111'));
// Mettre à jour la liste des références des locations de client 102
insert into table (select c.locations from clients c where c.numero='102')
(select ref(l) from locations l where l.numloc='1058');
```

6. Ecrire en SQL3 les requêtes suivantes :
   a. Quelles sont les agences (nom) qui loue des voitures de modèle PEUGEOT NEW 208?

```
select distinct deref(v.gere_par).nom
from vehicules v
where treat(value(v) as t_vehicule_immatricule).modele='PEUGEOT  NEW 208';
```

   b. Quelles sont les clients qui ont loué au moins une fois le même véhicule. Afficher des couples formés des noms des clients.

```
Select distinct c1.nom, c2.nom, treat(deref(value(l1).vehicule as t_vehicule_immatricule)).modele,
treat(deref(value(l2).vehicule as t_vehicule_immatricule)).modele
from clients c1, table(c1.locations) l1, clients c2, table(c2.locations)l2
where c1.numero<c2.numero
and (deref(deref(l1).vehicule).modele=deref(deref(l2).vehicule).modele);
```

   c. Quels sont les clients qui ont loué tous les véhicules de l'agence « Azimut Car » ?

```
select distinct c.nom
from clients c,  table(c.locations) l
where deref(deref(value(l).vehicule).gere_par).nom='Azimut Car'
having count(distinct(deref(value(l).vehicule).numveh))=(select count(*)
                                from vehicules v
                                where deref(v.gere_par).nom='Azimut Car')
group by c.nom;
```

7. On veut ajouter à ce schéma la méthode **Réviser** qui met à jour la date de la prochaine révision (attribut DateProchRev) à effectuer sur un véhicule. Les voitures doivent être révisées tous les 20000km, les motos tous les 6 mois et les vélos une fois par an.
   Définir la méthode **Réviser** (Signature et Corps) pour chaque type.

```
alter type t_vehicule add member procedure reviser cascade;
alter type t_moto add OVERRIDING member procedure reviser cascade;
create or replace type body t_moto
as OVERRIDING member procedure reviser
as
Begin
self.dateprochrev:=ADD_MONTHS(self.datedernierерev,6);
End ;
End;
/
alter type t_velo add OVERRIDING member procedure reviser cascade;
create or replace type body t_velo
as OVERRIDING member procedure reviser
as
Begin
self.dateprochrev:=ADD_MONTHS(self.datedernierерev,12);
End ;
End;
/
alter type t_voiture add OVERRIDING member procedure reviser cascade;
create or replace type body t_voiture
as OVERRIDING member procedure reviser
as
Begin
if (self.kilometrage-self.kilometragedernierерev)=20000)
then self.dateprochrev:=sysdate;
End ;
End;
/
```

NB: OVERRIDING pour redéfinir une méthode