

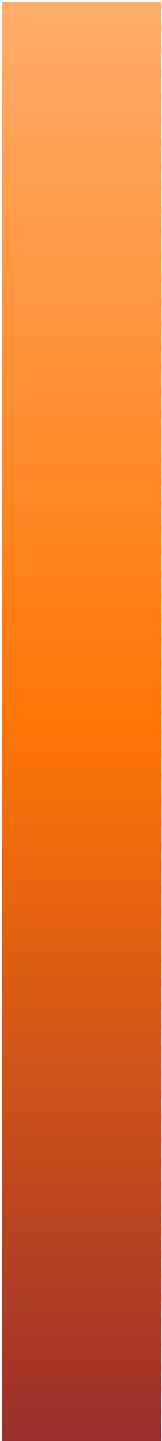
# Implémentation SQL de l'Héritage

**André Miralles**

# Implémentation de l'*Héritage* en BD

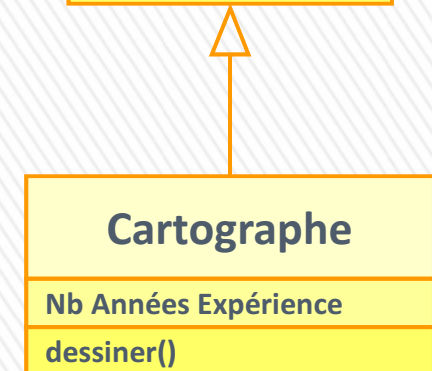
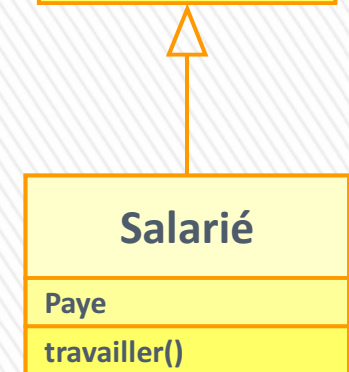
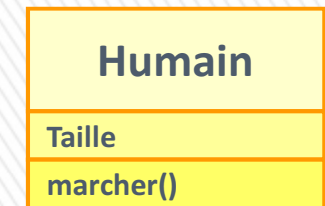
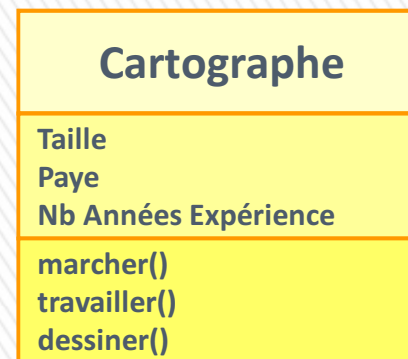
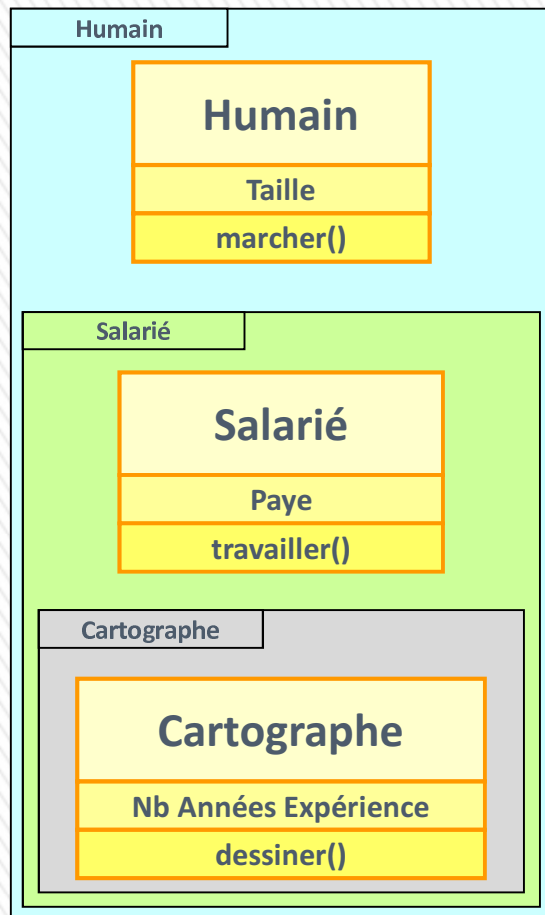
- » **Rappel du concept UML de la relation de Généralisation/Spécialisation**
  - > **Vision ensembliste**
  
- » **Relation de Généralisation/Spécialisation en SQL**
  - > **Base de données Relationnelle**
  - > **Base de données Objet**





# Rappel du concept UML de la relation de Généralisation / Spécialisation

# Concept de *Généralisation/Spécialisation*



## » Spécialisation

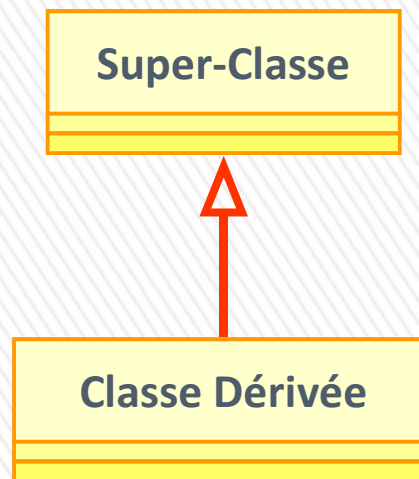
- > **Mécanisme** permettant de définir une **classe fille** comme étant un **sous-ensemble** d'une **classe mère**



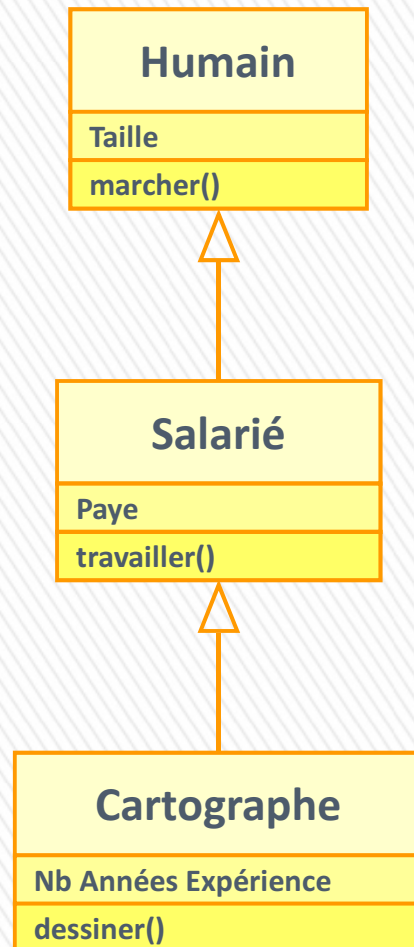
# Concept de *Généralisation/Spécialisation*

## >> Notation

- > **Flèche à extrémité triangulaire** orientée de la **classe dérivée** (classe fille) vers la **super-classe** (classe mère)



# Concept de *Généralisation/Spécialisation*



## » Spécialisation

### > La classe fille

- + **Hérite** des **attributs**, des **opérations** et des **associations** de la classe mère
- + **Ajoute** des attributs, des opérations et des associations **propres**
- + **Peut redéfinir** le sens des attributs et des opérations de la **classe mère** (à manipuler avec beaucoup de précaution)

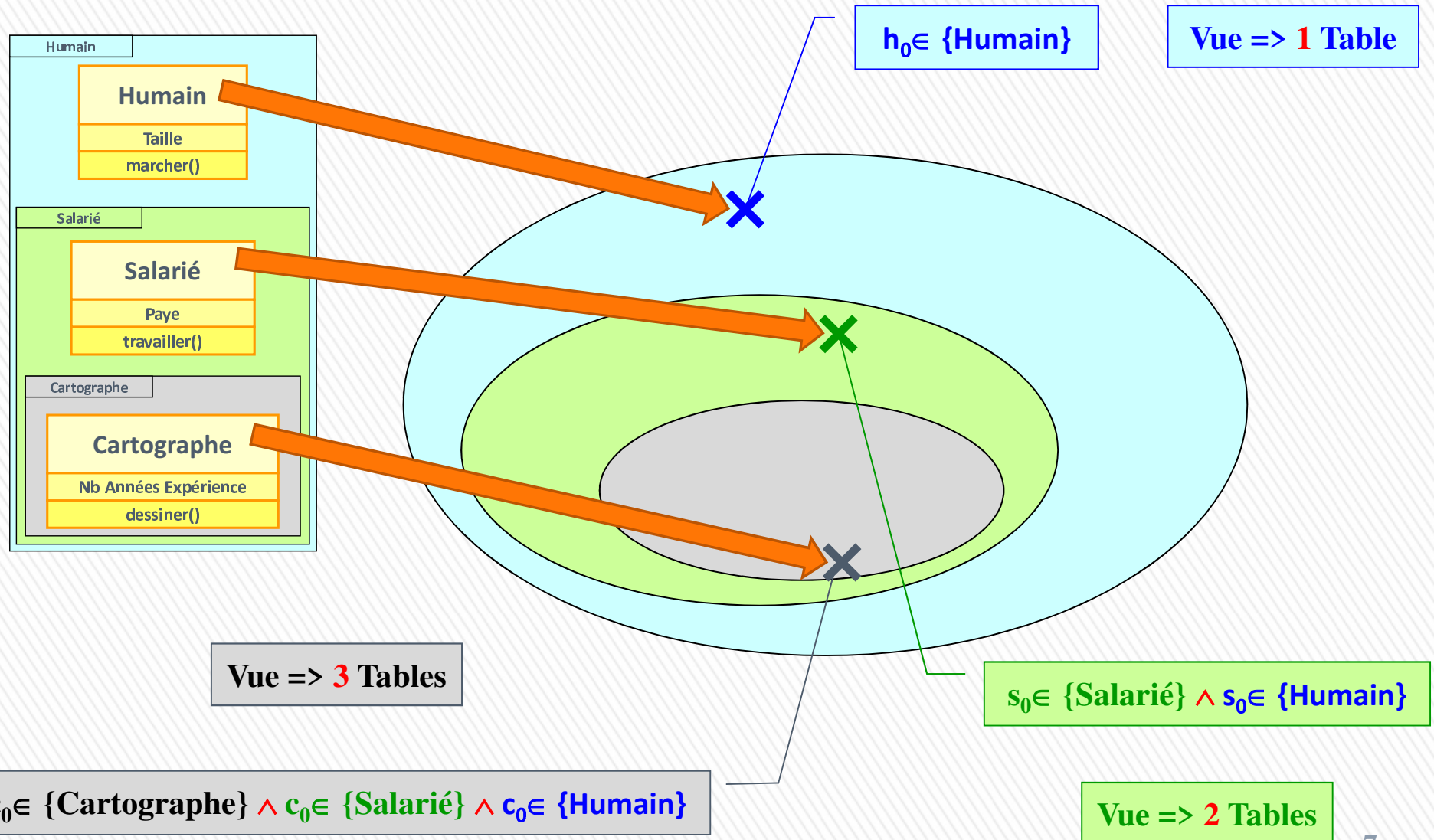
## » Généralisation

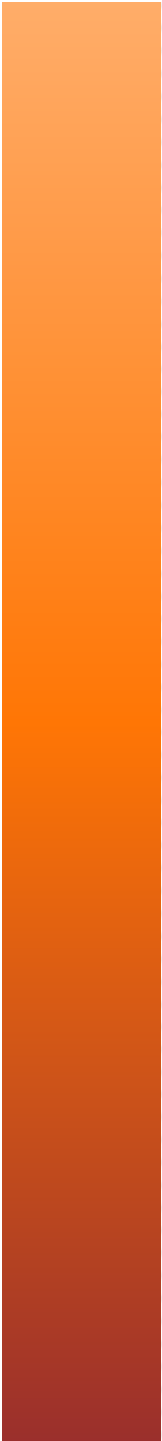
- > Permet de **Fédérer/Mutualiser** des attributs, des opérations et des associations communs aux **classes filles**



# Concept de *Généralisation/Spécialisation*

## » Vision ensembliste





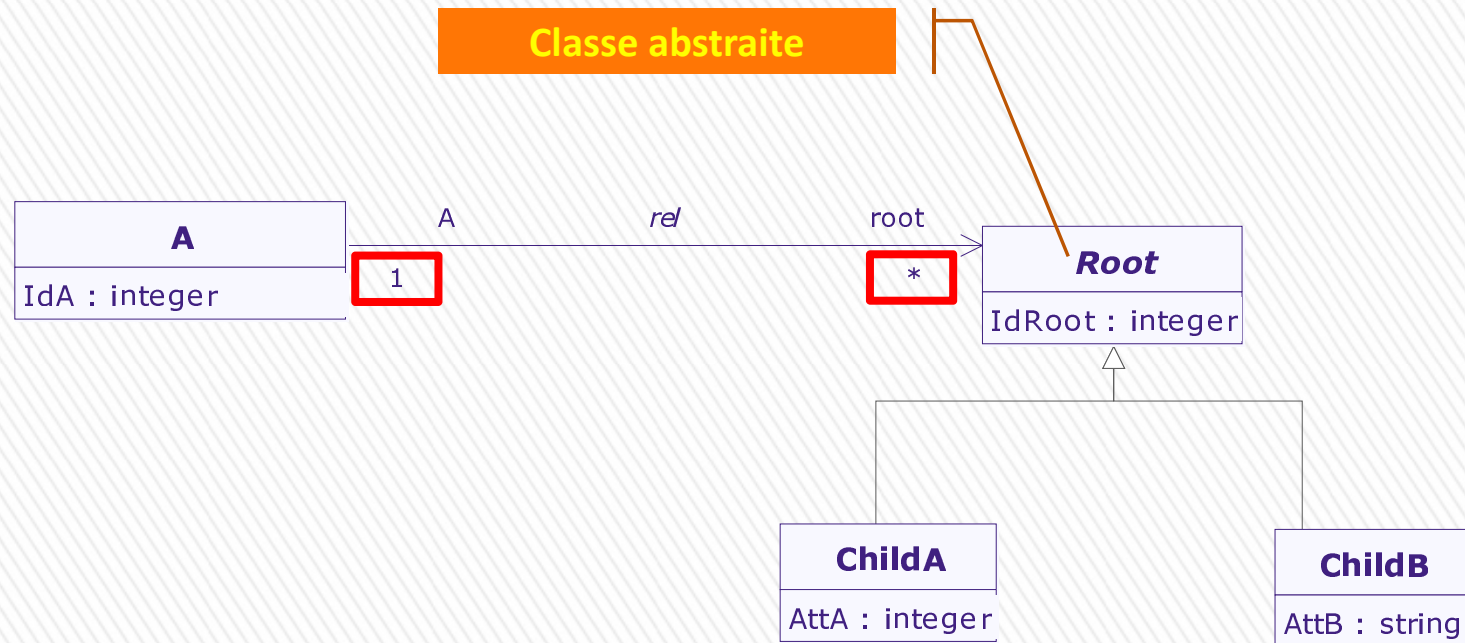
# Relation de Généralisation / Spécialisation en SQL

**Base de données  
Relationnelle**



# Base de données *Relationnelle*

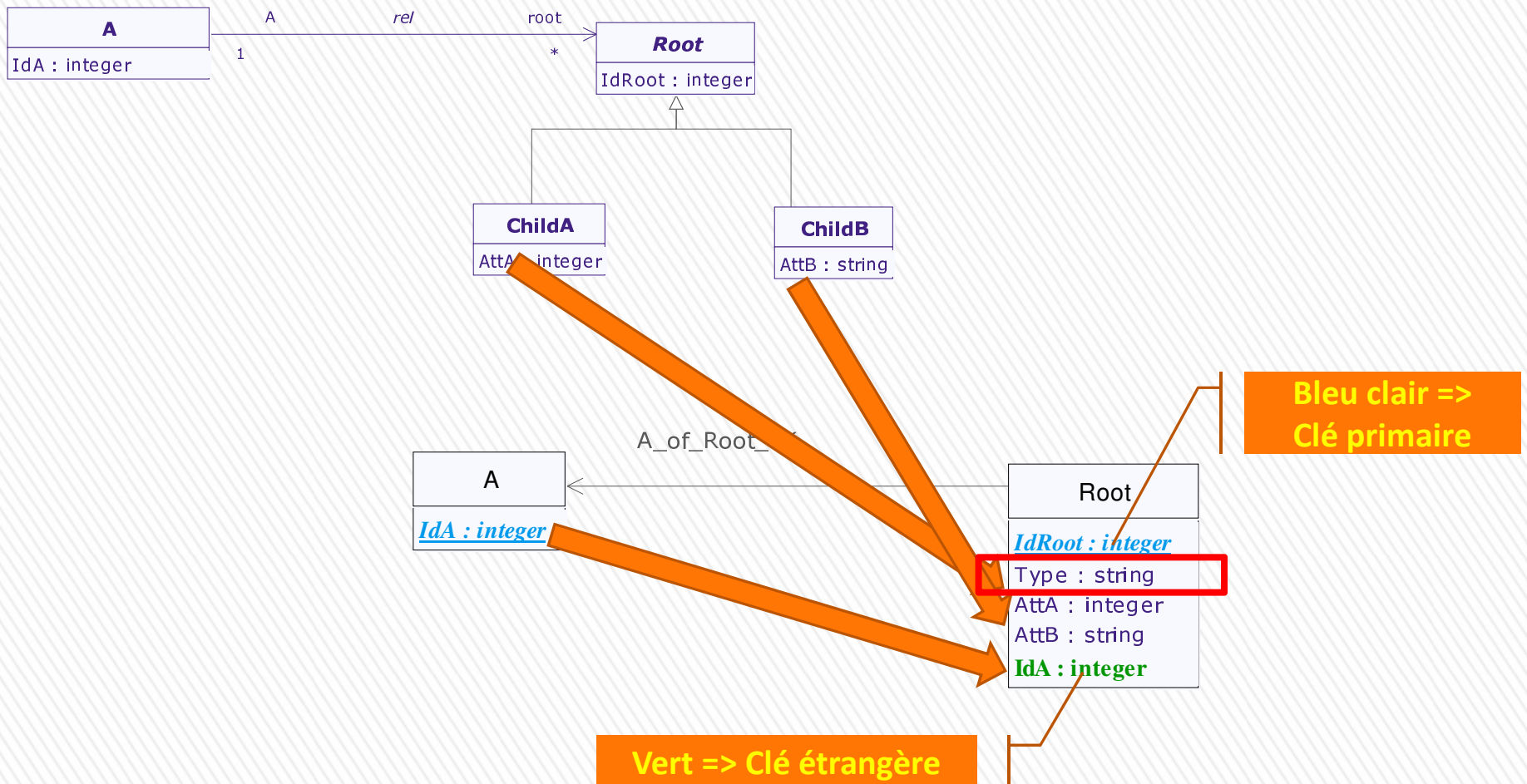
## » Modèle d'analyse **enrichi des identifiants**



# Base de données *Relationnelle*

## » Modèle physique des données

> Héritage => **Table UNIQUE**





# Base de données *Relationnelle*

## » Héritage => Table UNIQUE

> CREATE TABLE A(

+ IdA INTEGER NOT NULL,

+ CONSTRAINT A\_PK PRIMARY KEY (IdA));

> CREATE TABLE Root(

+ IdRoot INTEGER NOT NULL ,

+ Type VARCHAR(50),

+ AttA INTEGER,

+ AttB VARCHAR(50),

+ IdA INTEGER NOT NULL,

+ CONSTRAINT Root\_PK PRIMARY KEY (IdRoot),

+ CONSTRAINT A\_to\_Root\_FK FOREIGN KEY (IdA) REFERENCES A(IdA)),

+ CONSTRAINT Type\_DK Type IN ('Root', 'ChildA', 'ChildB'));

Classe abstraite

Liste des noms des tables

# Base de données *Relationnelle*

» Héritage => **Table UNIQUE**

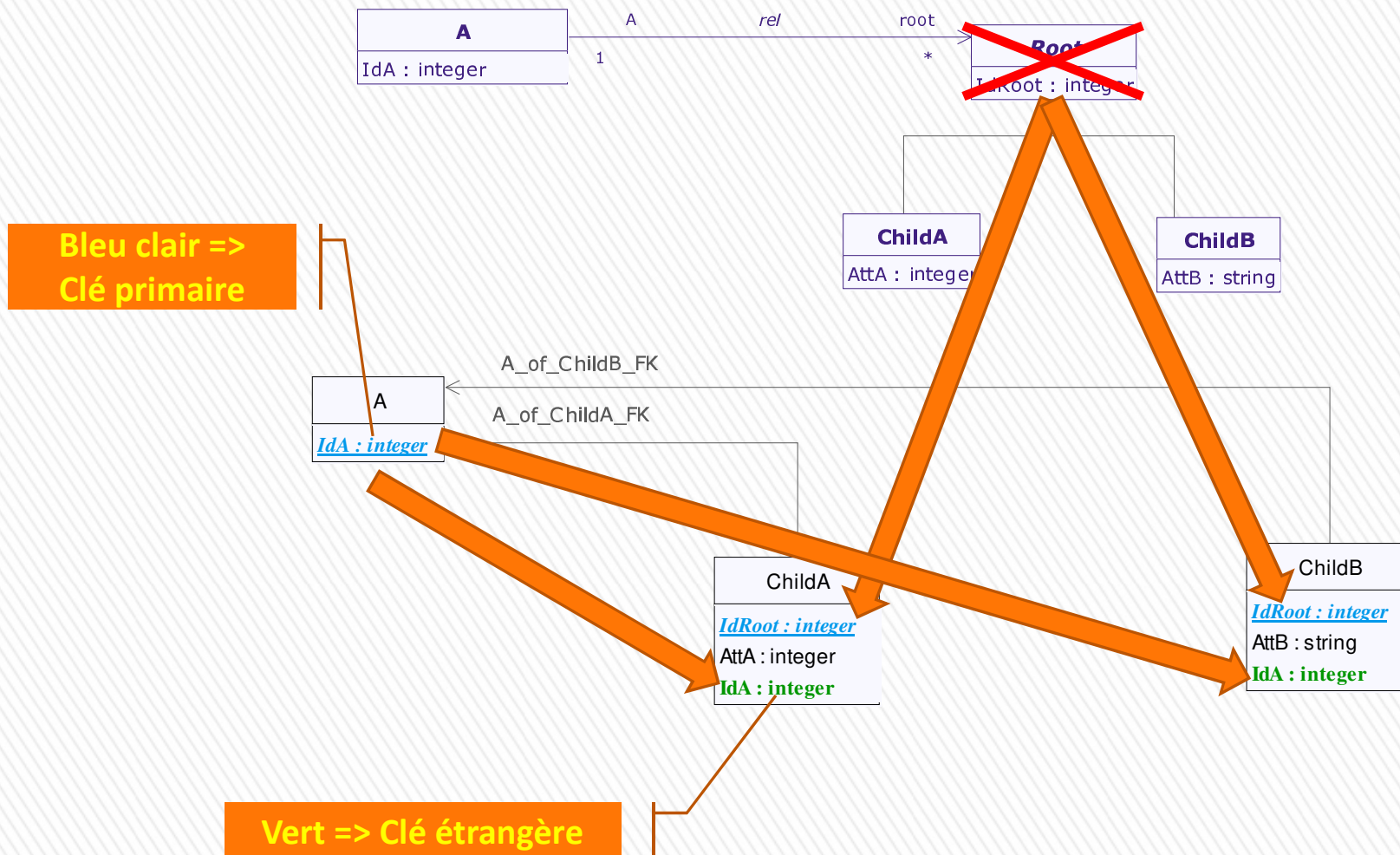
**Inconvénient => Beaucoup de valeurs NULLES**



# Base de données *Relationnelle*

## » Modèle physique des données

> Héritage => **UNE table par classe SAUF les classes ABSTRAITES**



# Base de données *Relationnelle*

» Héritage => **UNE table par classe SAUF les classes ABSTRAITES**

> CREATE TABLE **A**(

+ **IdA** INTEGER NOT NULL,

+ **CONSTRAINT A\_PK** PRIMARY KEY (IdA));

> CREATE TABLE **ChildA**(

+ **IdRoot** INTEGER NOT NULL,

+ **AttA** INTEGER,

+ **IdA** INTEGER NOT NULL,

+ **CONSTRAINT ChildA\_PK** PRIMARY KEY (IdRoot),

+ **CONSTRAINT A\_to\_ChildA\_FK** FOREIGN KEY (IdA) REFERENCES A(IdA));

> CREATE TABLE **ChildB**(

+ **IdRoot** INTEGER NOT NULL,

+ **AttB** VARCHAR(255),

+ **IdA** INTEGER NOT NULL,

+ **CONSTRAINT ChildB\_PK** PRIMARY KEY (IdRoot),

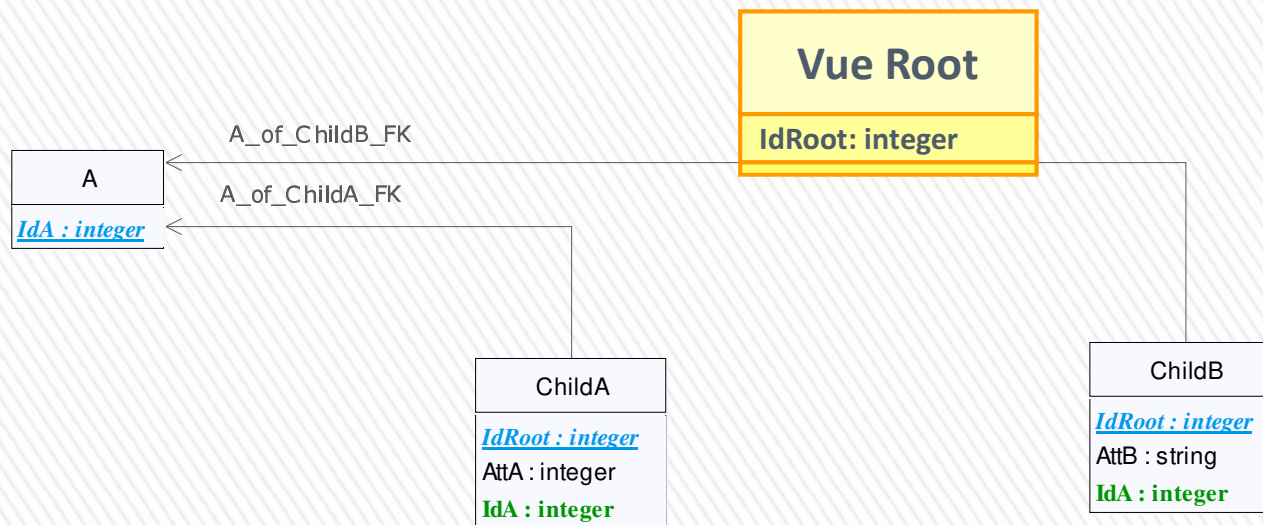
+ **CONSTRAINT A\_to\_ChildB\_FK** FOREIGN KEY (IdA) REFERENCES A(IdA));



# Base de données *Relationnelle*

» Héritage => **UNE table par classe SAUF les classes ABSTRAITES**

**Inconvénient => Difficulté de gestion de l'UNICITÉ des IDENTIFIANTS**

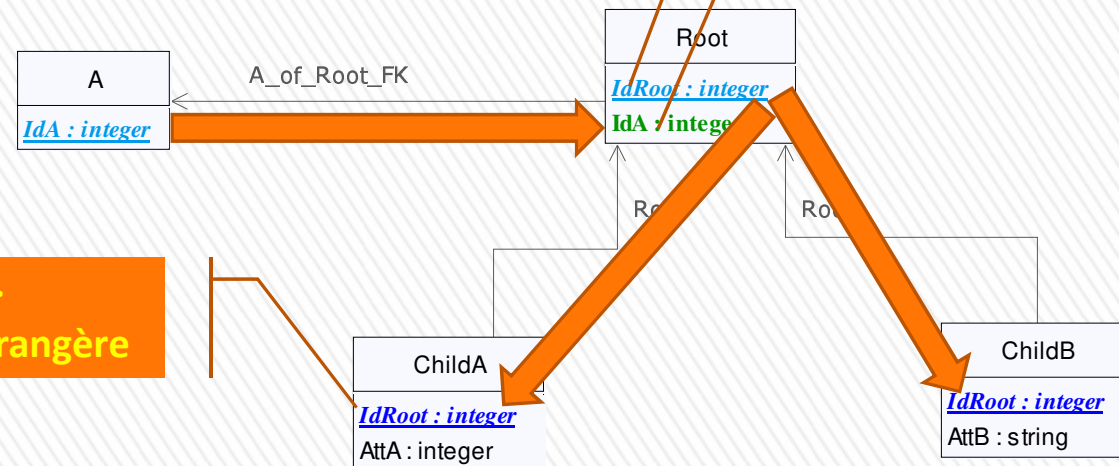
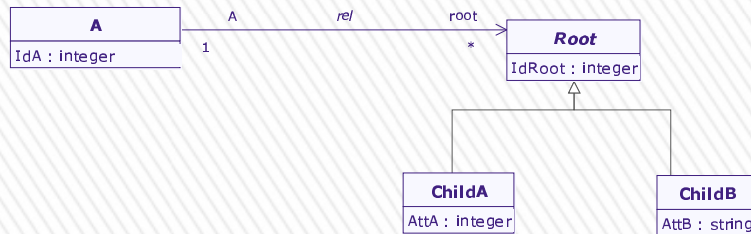


**Solution => Création couple VUE + TRIGGER**

# Base de données *Relationnelle*

## » Modèle physique des données

### > Héritage => **UNE table par classe**



Bleu clair =>  
Clé primaire

Vert => Clé étrangère

Bleu foncé =>  
Clé primaire + Clé étrangère



# Base de données *Relationnelle*

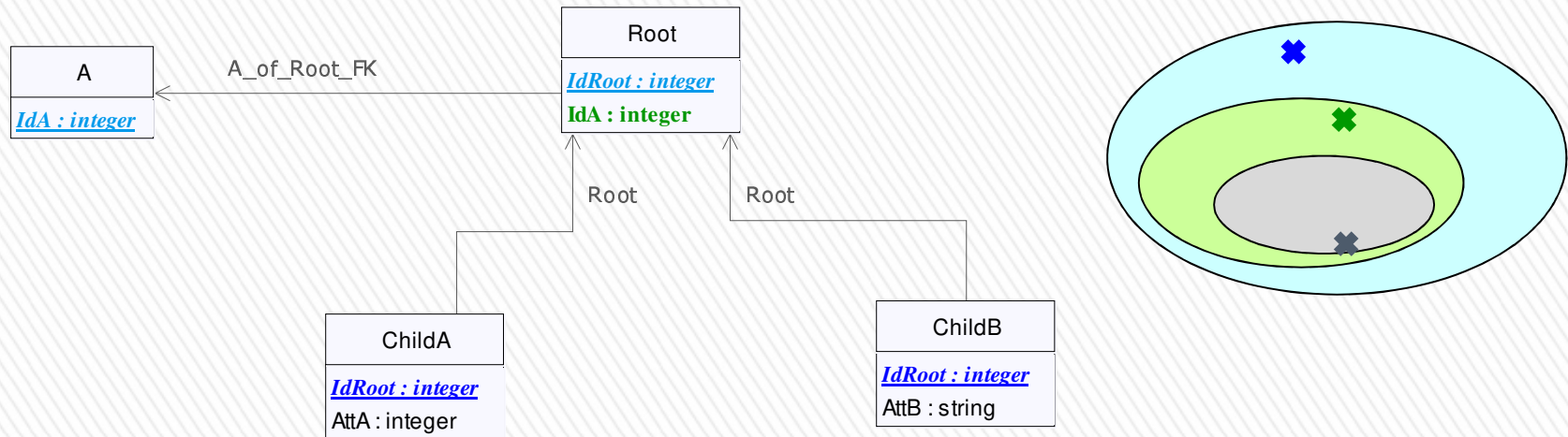
## » Héritage => **UNE table par classe**

```
> CREATE TABLE A(  
  + IdA INTEGER NOT NULL,  
  + CONSTRAINT A_PK PRIMARY KEY (IdA));  
  
> CREATE TABLE Root(  
  + IdRoot INTEGER NOT NULL,  
  + IdA INTEGER NOT NULL,  
  + CONSTRAINT Root_PK PRIMARY KEY (IdRoot),  
  + CONSTRAINT A_to_Root_FK FOREIGN KEY (IdA) REFERENCES A(IdA));  
  
> CREATE TABLE ChildA(  
  + IdRoot INTEGER NOT NULL,  
  + AttA INTEGER,  
  + CONSTRAINT ChildA_PK PRIMARY KEY (IdRoot),  
  + CONSTRAINT Root_to_ChildA_FK FOREIGN KEY (IdRoot) REFERENCES Root(IdRoot));  
  
> CREATE TABLE ChildB(  
  + IdRoot INTEGER NOT NULL,  
  + AttB VARCHAR(255),  
  + CONSTRAINT ChildB_PK PRIMARY KEY (IdRoot),  
  + CONSTRAINT Root_to_ChildB_FK FOREIGN KEY (IdRoot) REFERENCES Root(IdRoot));
```

# Base de données *Relationnelle*

» Héritage => **UNE table par classe**

**Attention => Réplication des identifiants dans TOUTES les tables « mères »**



**Avantage => UNICITÉ des identifiants assurée**



# Base de données *Relationnelle*

## » **Avantage**

- > Solution implémentable dans **tous les SGBD**
- > Consommation de **ressources faible** puisqu'on est en relationnel

## » **Inconvénient**

- > Conflit avec le framework **Hibernate**

## » **Conseil**

- > Au vu des avantages, implémenter l'option **UNE table par classe**

# Relation de Généralisation / Spécialisation en SQL

**Base de données**  
**Objet PostgreSQL**



# Base de données *Objet*

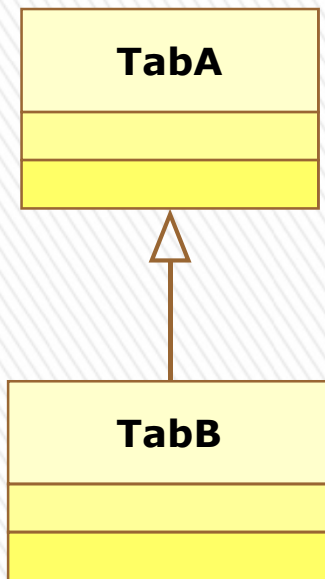
## » Héritage

- > Introduction en **SQL 3** (Standardisation en 1999 « SQL99 »)
- > Implémentation **différente** suivant le SGBD

## » PostgreSQL => **Héritage de tables**

- > Mécanisme d'héritage entre tables
- > Héritage **multiple possible**

# Implémente de l'héritage entre tables via l'instruction **INHERITS**



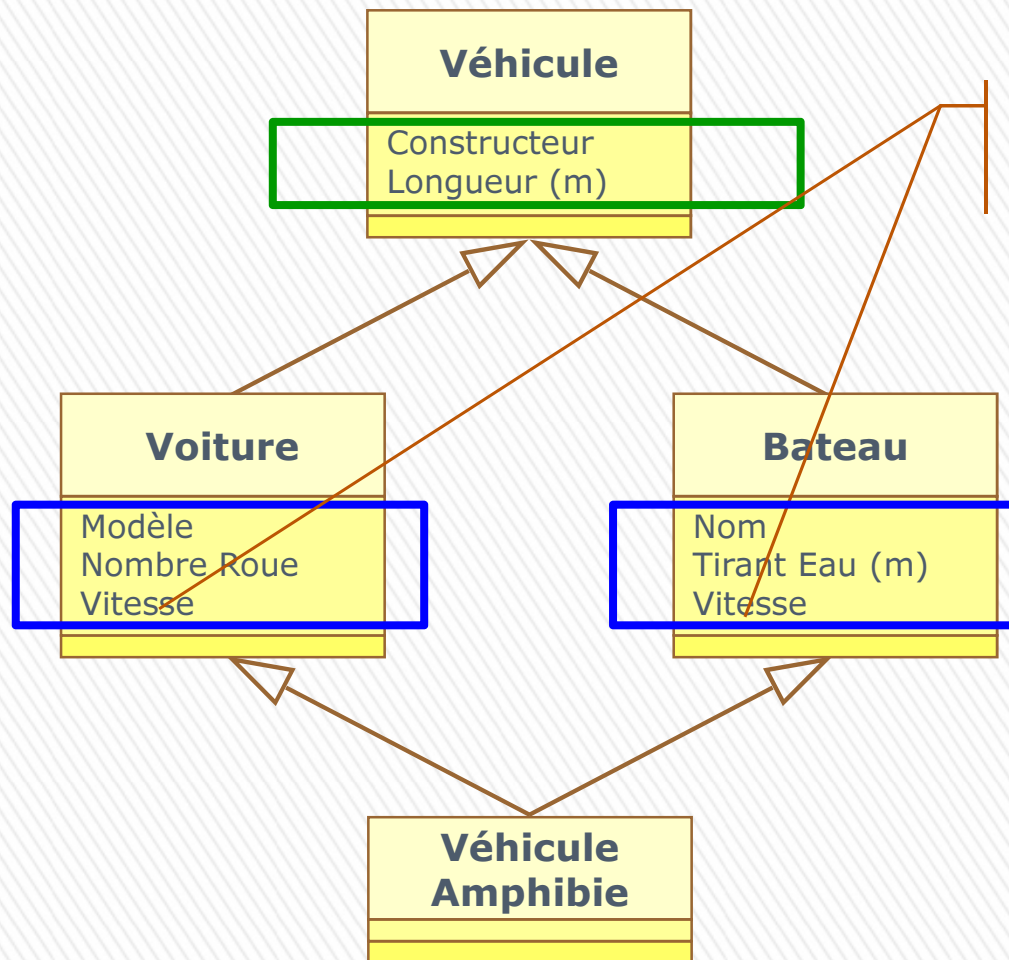
## » Exemple de code

```
> CREATE TABLE tabA (  
+ .....  
+ );
```

```
> CREATE TABLE TabB (  
+ .....  
+ ) INHERITS (TabA);
```

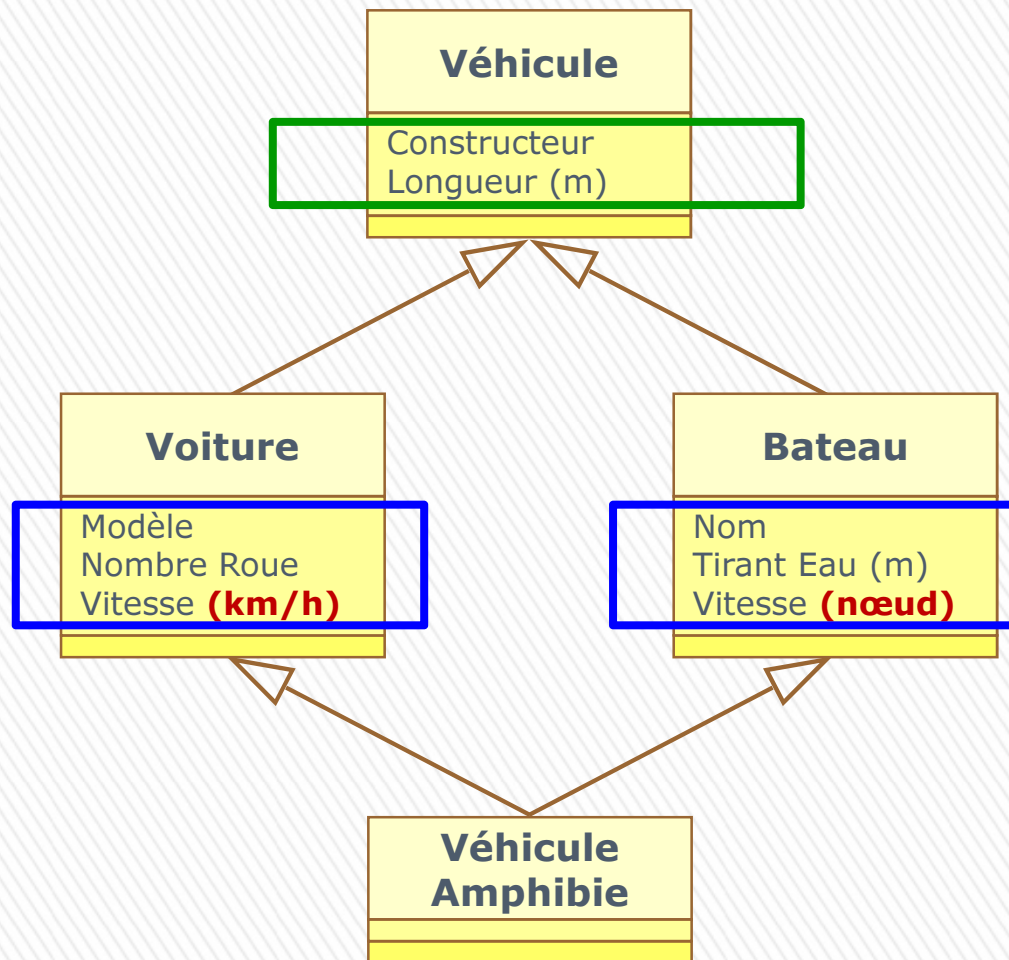


# Exemple du *Véhicule Amphibie*



Est-ce qu'il faut déplacer la vitesse dans le classe Mère ?

# Exemple du *Véhicule Amphibie*





# Exemple du *Véhicule Amphibie*

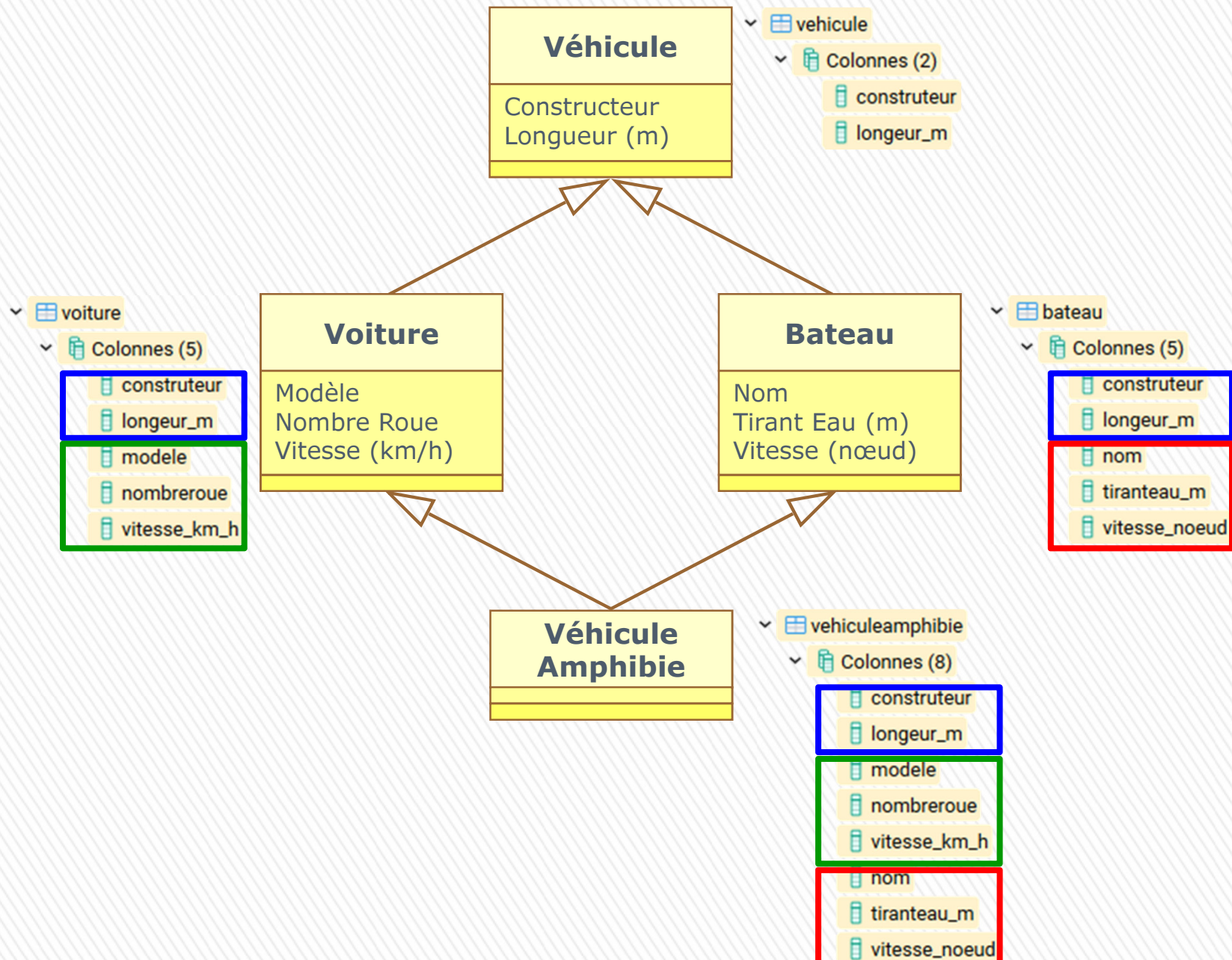
```
» CREATE TABLE Vehicule(  
    Constructeur varchar,  
    Longueur_M real  
);
```

```
» CREATE TABLE Voiture(  
    Modele varchar,  
    NombreRoue integer,  
    Vitesse_KM_H real  
    ) INHERITS(Vehicule);
```

```
» CREATE TABLE Bateau(  
    Nom varchar,  
    TirantEau_M real,  
    Vitesse_Noeud real  
    ) INHERITS(Vehicule);
```

```
» CREATE TABLE VehiculeAmphibie(  
    ) INHERITS(Voiture,Bateau);
```

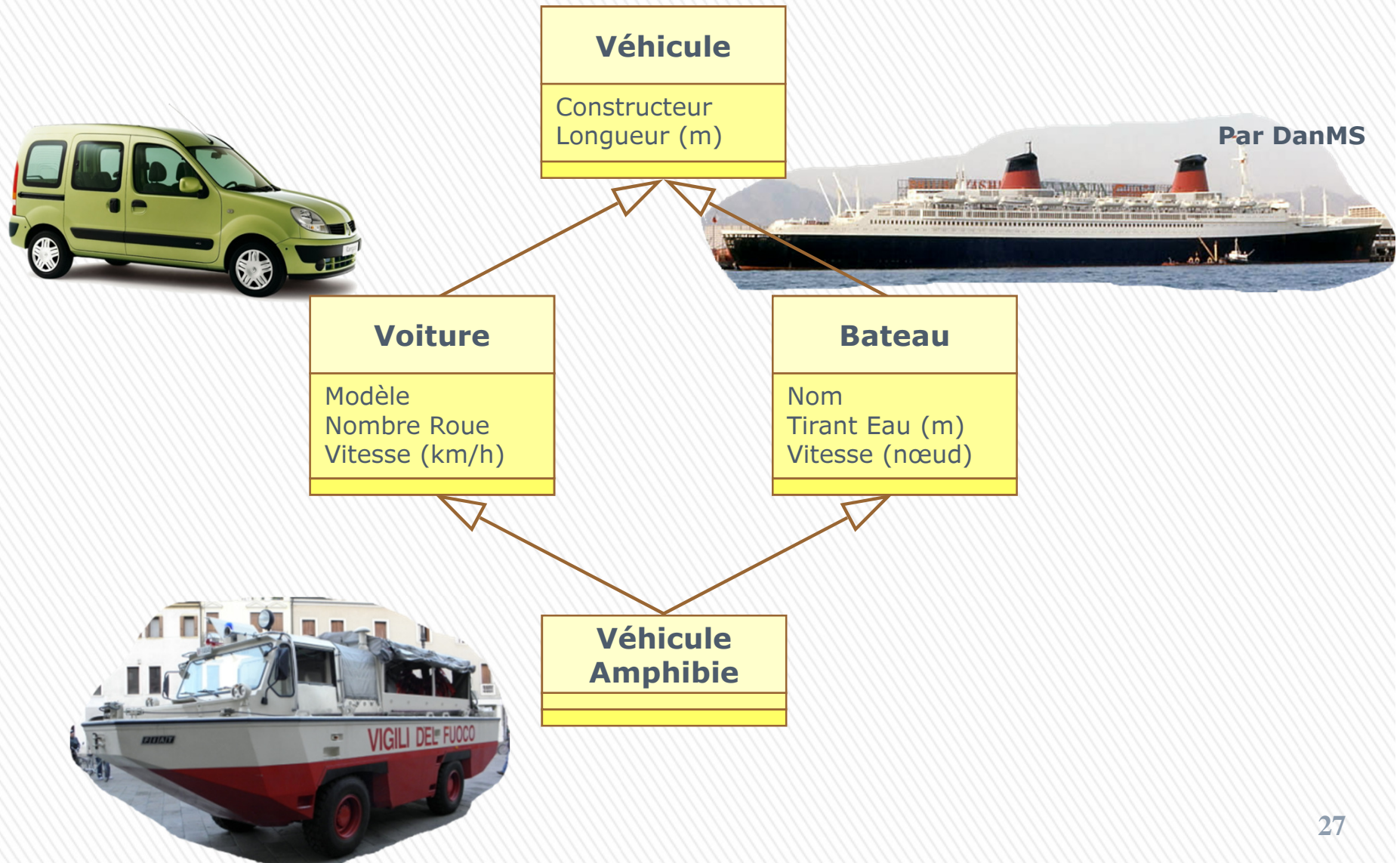
# Héritage de Table dans PostgreSQL





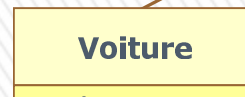
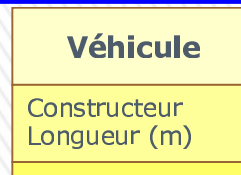
# Exemple du *Véhicule Amphibie*

## Insertion de données



# Exemple du *Véhicule Amphibie*

Données		
	construteur	longeur_m
	character varying	real
1	Renault	4.282
2	Chantiers de l'Atlantique	315.7
3	Fiat	7.3



Données				
	construteur	longeur_m	modele	nombreuroues
	character varying	real	character varying	integer
1	Renault	4.282	Kangoo	4
2	Fiat	7.3	FIAT 6640 A	4

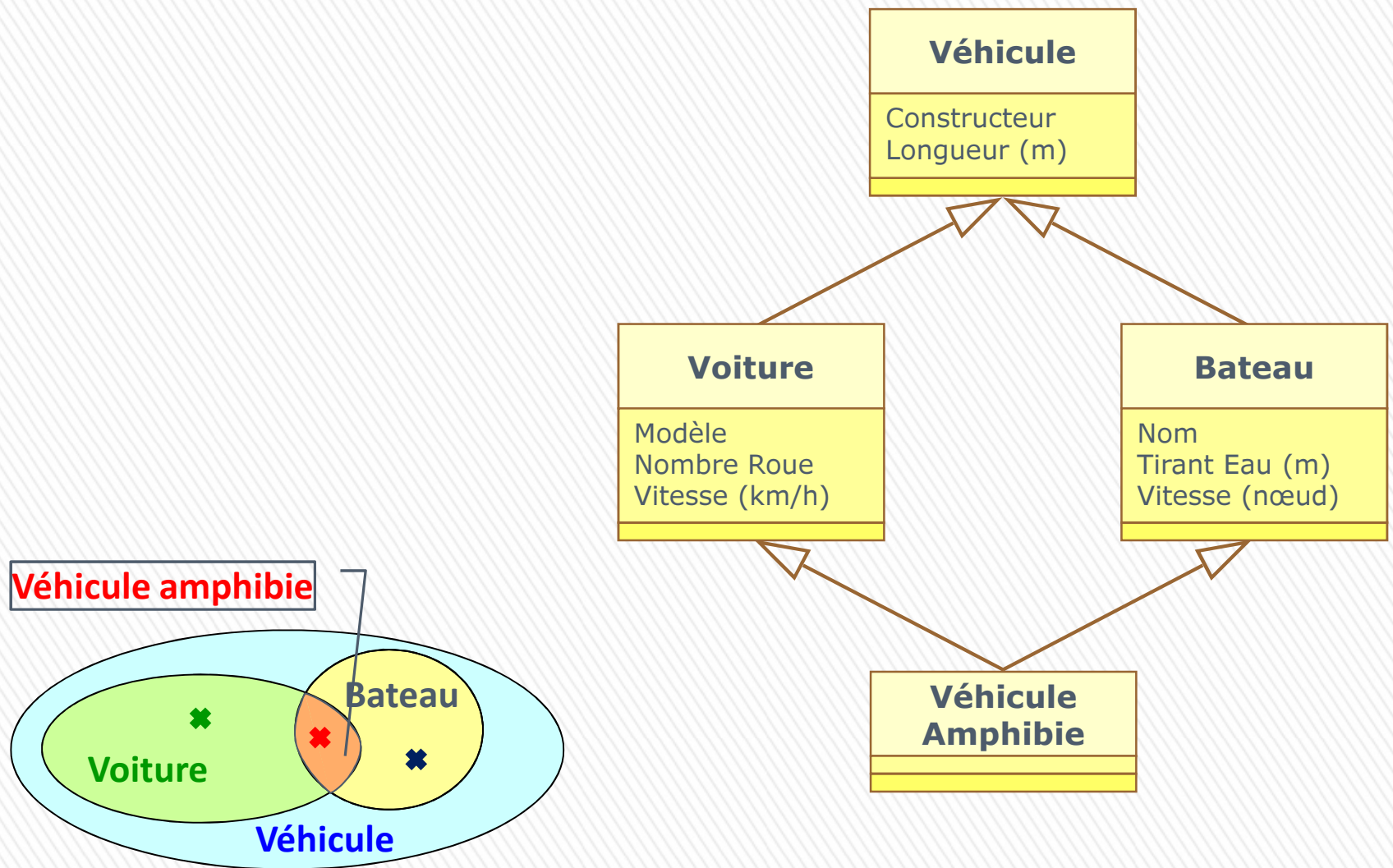
Données				
	construteur	longeur_m	nom	tiranteau_m
	character varying	real	character varying	real
1	Chantiers de l'Atlantique	315.7	Le France	10.5
2	Fiat	7.3		0.5



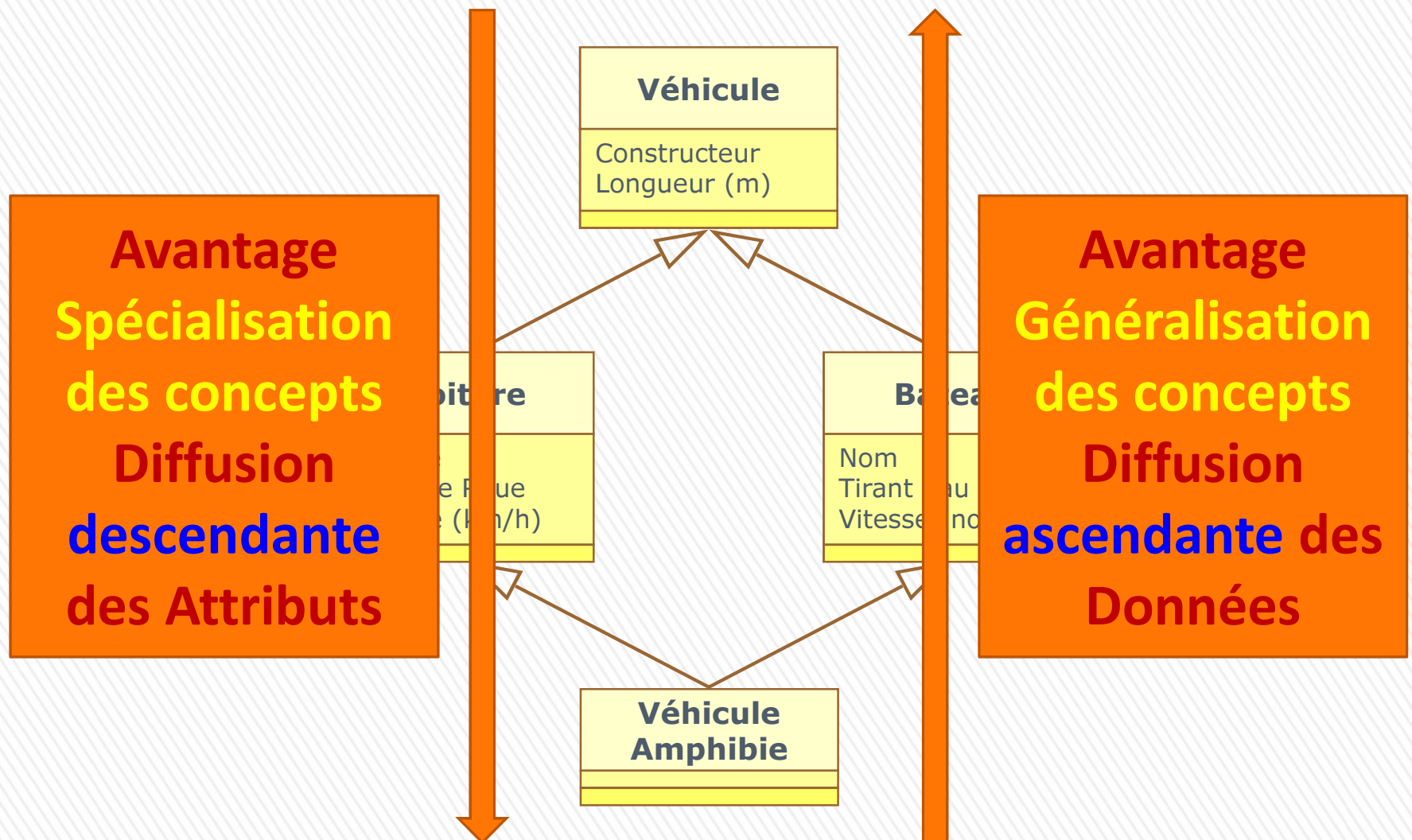
Données							
	construteur	longeur_m	modele	nombreuroues	vitesse_km_h	nom	tiranteau_m
	character varying	real	character varying	integer	real	character varying	real
1	Fiat	7.3	FIAT 6640 A	4	100		0.5



# Héritage de Table dans PostgreSQL



# *Héritage de Table* dans PostgreSQL



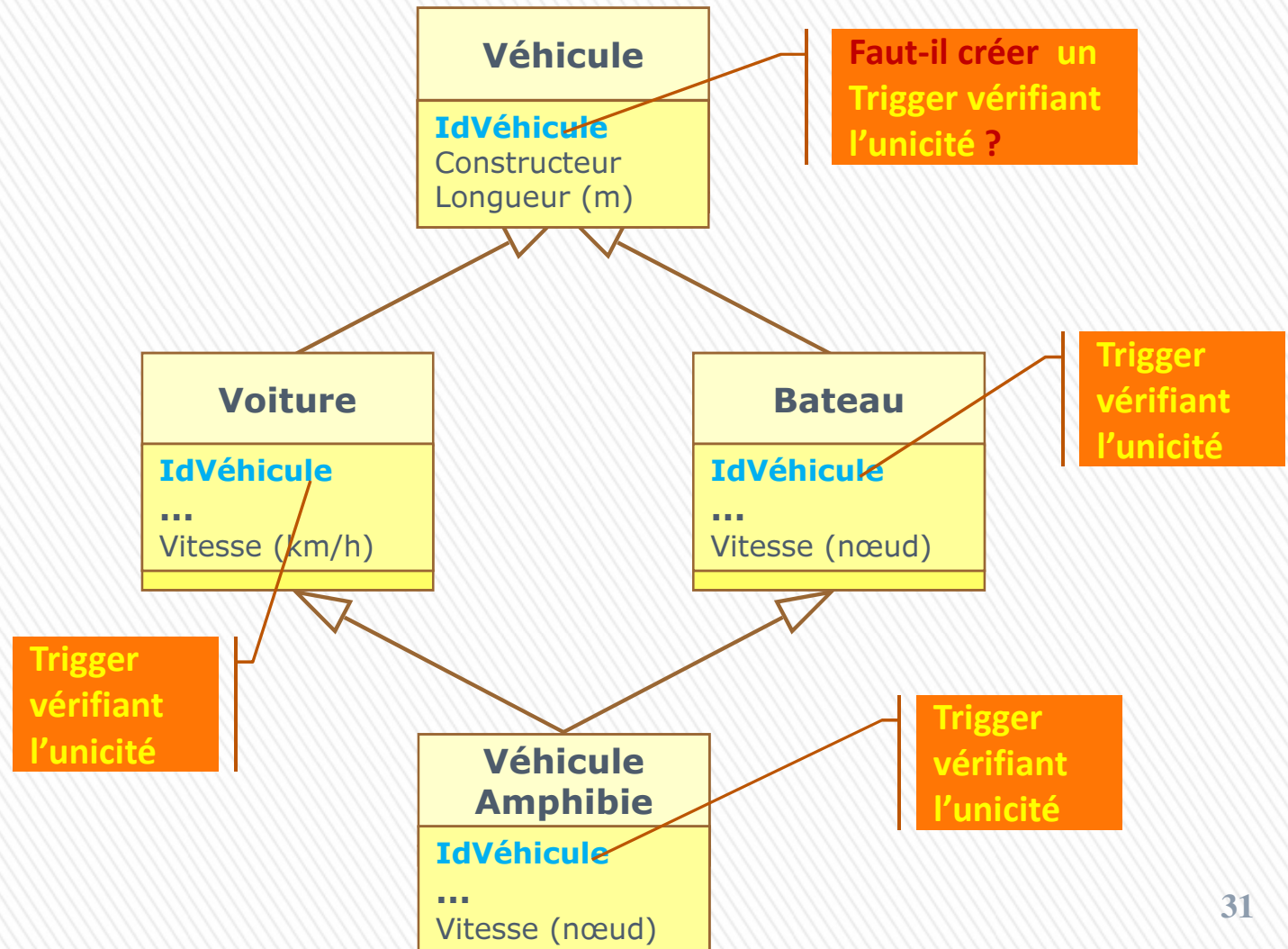


# Unicité de *Identifiants*

## Pour des classes *non abstraites*

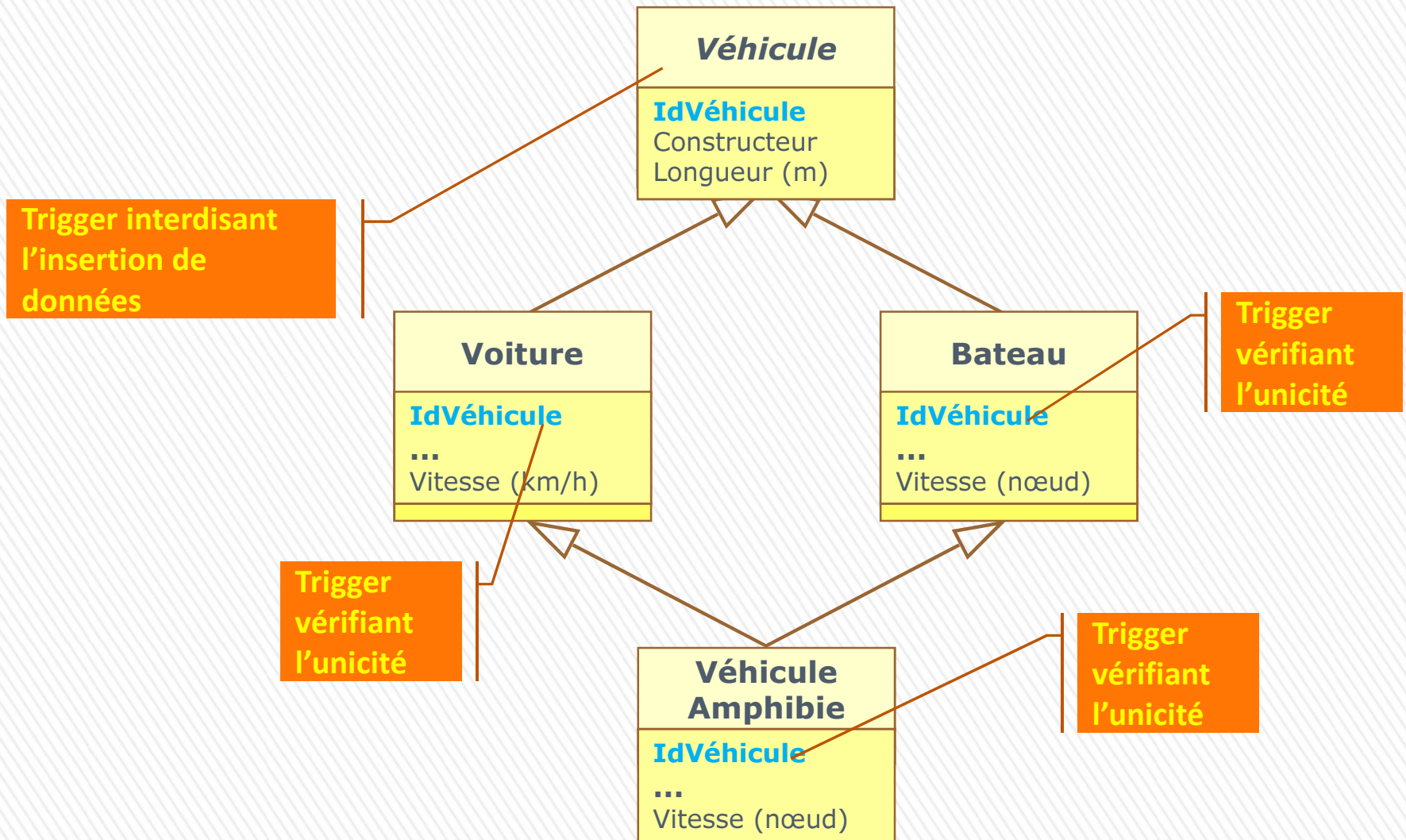
**Pb => Unicité de identifiants non assurée**

**Pb**  
Pas de  
Diffusion  
de la  
contrainte  
de clé  
Primaire



# Unicité de *Identifiants*

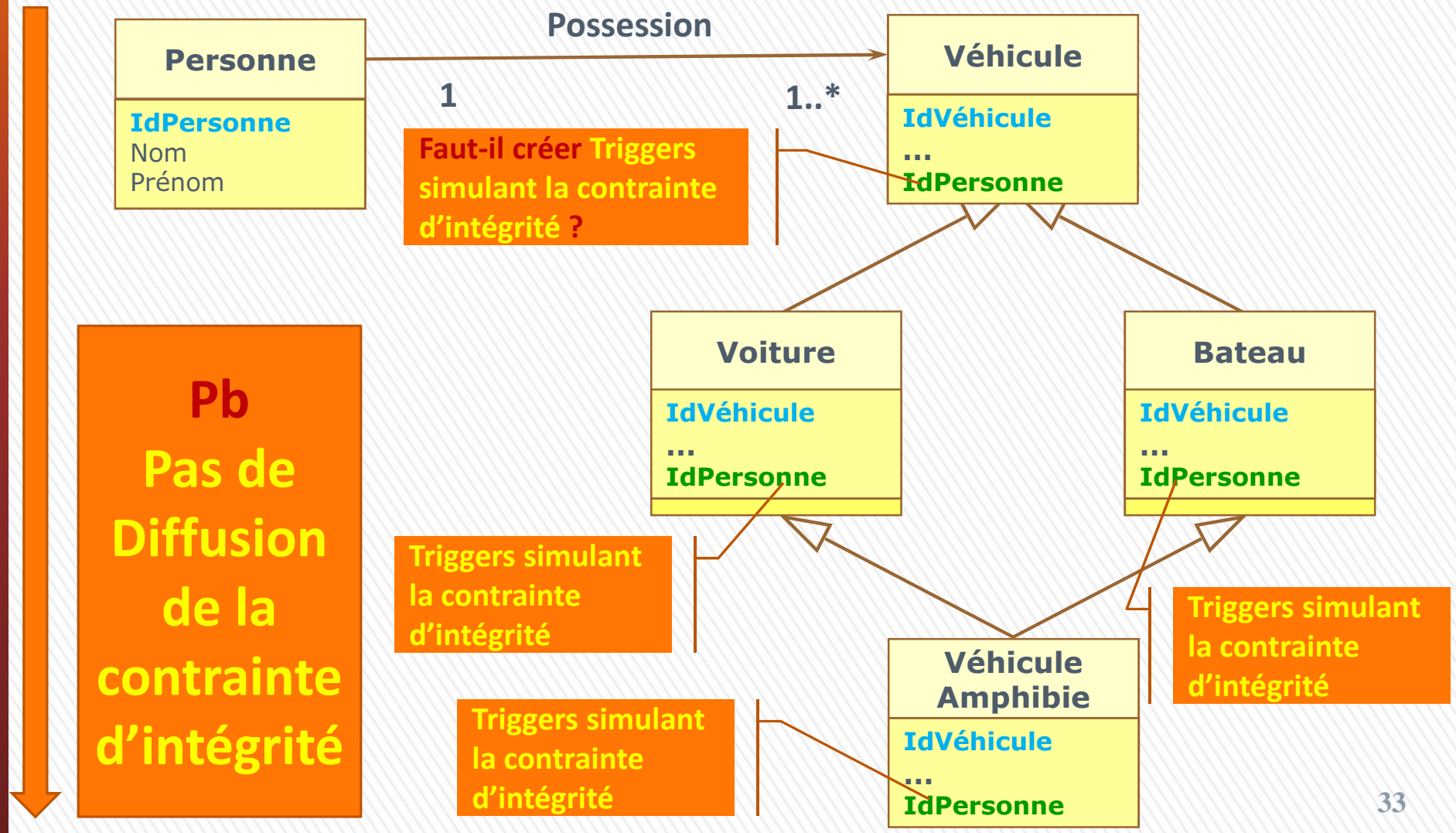
## *Classe mère abstraite*





# Contraintes d'intégrité

**Pb => Intégrité des données non assurée**



# En conclusion

**Muni des Triggers => Base de données est cohérente**

