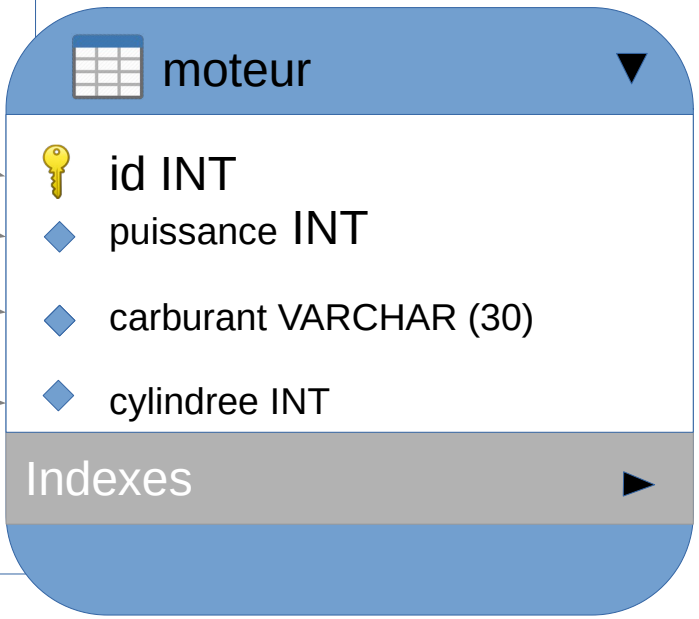
@OneToOne – java config

Suivi des étapes

- Préparer les tables coté BDD (créer les relations entre les tables)
- Coder les classes en entities en java
- Créer une classe executable pour manipuler les entités

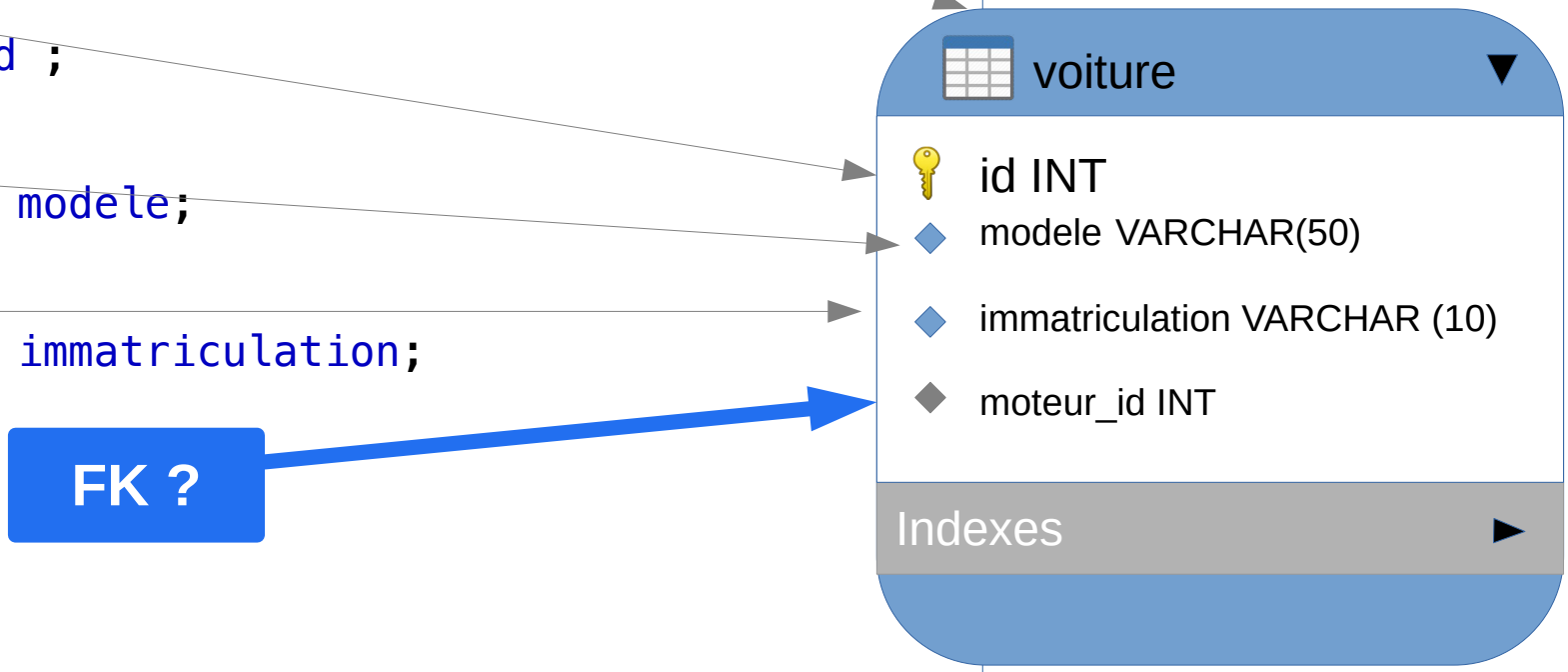
Créer la classes d'entité Moteur

```java
@Entity
public class Moteur {

@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
@Column
private int id;

@Column
private int  puissance;

@Column
private String  carburant;

@Column
private int  cylindree;

//getters() & setters()
```



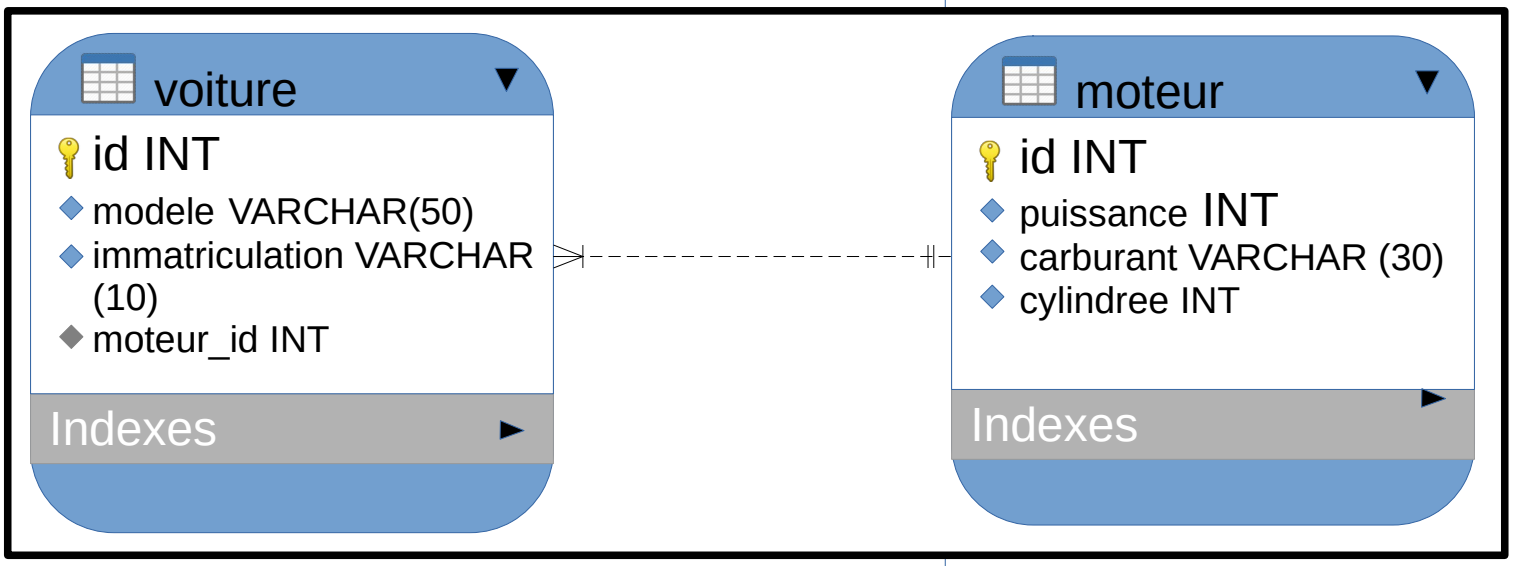
Créer la classe d'entité Voiture

```java
@Entity
@Table(name="voiture")
public class Voiture {

@Id
@GeneratedValue(strategy= GenerationType.IDENTITY)
@Column
private Long id ;

@Column
private String modele;

@Column
private String immatriculation;

//getters & setters
. . .
```



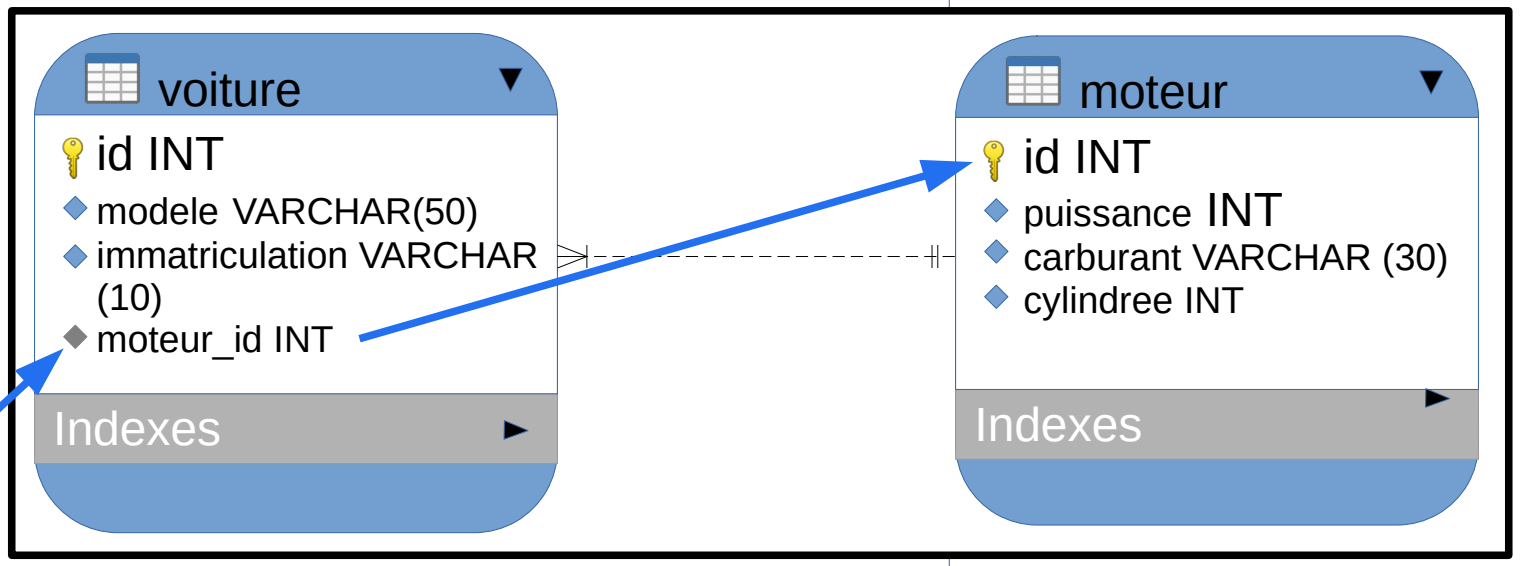
@OneToOne

```
@Entity
@Table(name="voiture")
public class Voiture {

...

@OneToOne
@JoinColumn(name="moteur_id")
private Moteur moteur;

//getters & setters
. . .
```



@OneToOne

```java
@Entity
@Table(name="voiture")
public class Voiture {

...

@OneToOne
@JoinColumn(name="moteur_id")
private Moteur moteur;

//getters & setters
. . .
```



Cycle de vie des entités

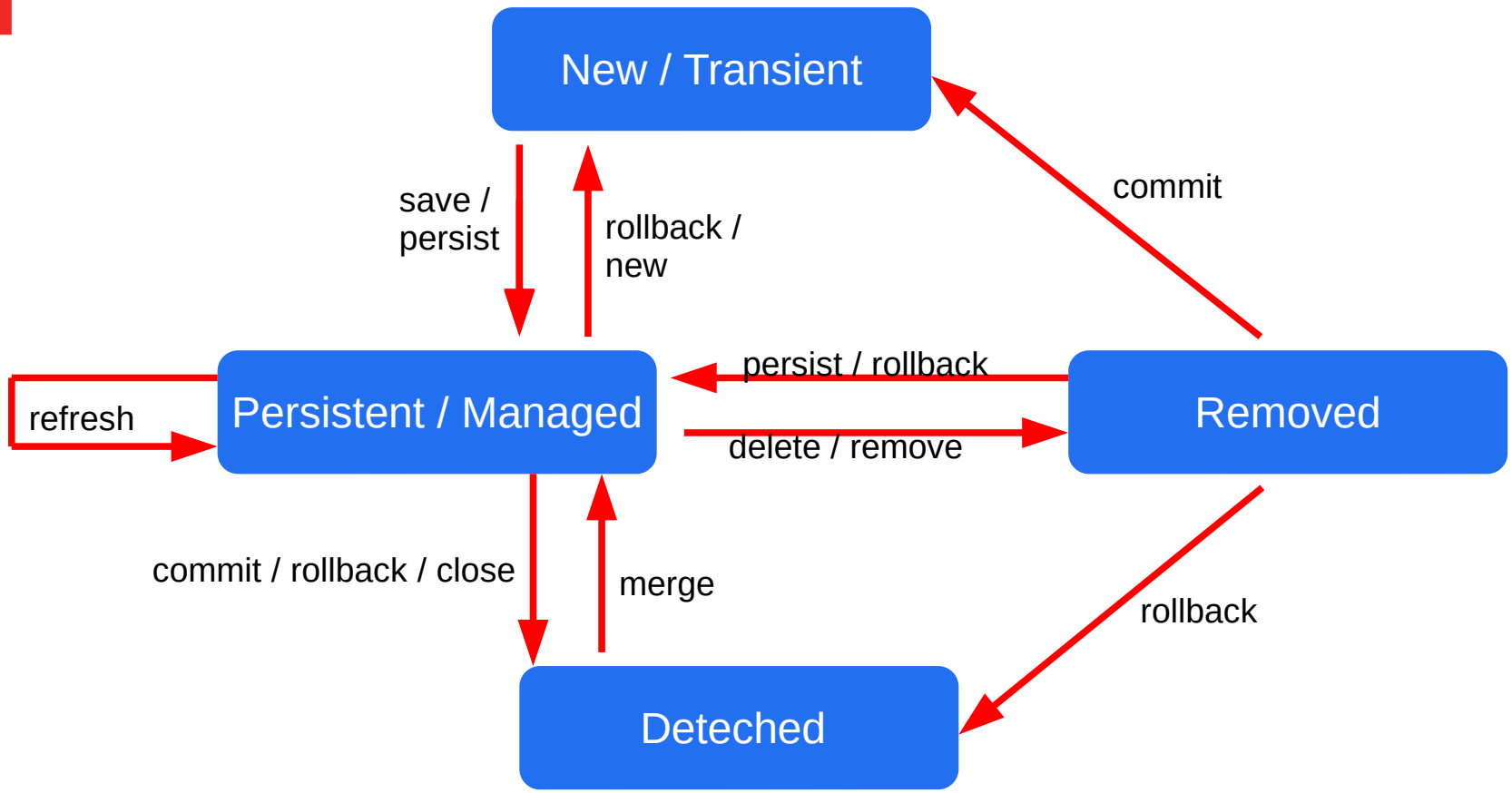
C'est hibernate qui considère une classe comme une entité à partir du moment ou cette classe est annotée @Entity et correctement mappée.

- Alors hibernate va considérer les instances de cette classe, et qualifer , préciser leur état vis à vis de ce qui existe en bdd et ce qui existe dans la mémoire java.
- Cet état va succéssivement évoluer au gré des opérations qu'on va effectuer avec hibernate sur ces instances.

Entity lifecycle

Opérations	Description
Detach	Décrit une entité qui n'est pas associée à une session hibernate
Merge	"merger" une instance est détachée c'est la fusionner , la rattacher à une session.
Persist	Pour une nouvelle instance , on va la gére, la passer dans un été géré (managed"), et le prochain commit elle sera ainsi sauvegardée
Remove	Une entité gérée est ainsi supprimée , au prochain flush() ou commit elle sera supprimée (deleted)
Refresh	Recharge ou rafraichit , synchronise l'instance , (écrase) l'objet avec les informations en bdd , évite d'avoir des données obsolète dans le code java

Diagrame : appels de méthodes de la session



Opérations en cascade

- Cascader l'opération de création sur un objet voiture va "en cascade" créer d'abord un objet Moteur et ensuite créer la un objet voiture pour pouvoir les relier via la FK
- Une suppression cascade fera la suppresssion de la voiture puis celle en cascade du moteur.
- L'ordre d'execution en cascade est dicté par les contraintes FK en SQL. Mais le principe d'opérations en cascade c'est que la même opération sera effectuée sur les deux objets.

@OneToOne & Cascade Types

Cascade Type	Description
PERSIST	Si l'entité est persitée, l'entité liée est persistée aussi
REMOVE	Si l'entité est supprimée, l'entité supprimée est supprimée
REFRESH	Si l'entité est synchronisée, l'entité liée est synchronisée
DETACH	Si l'entité est "detached" (non associée à une session) , alors l'entité liée sera également "detached"
MERGE	Si l'entité est fusionnée, alors l'entitée liée sera fusionnée aussi.
ALL	Toutes les types d'opérations sont cascadés.

Configurer le Cascade Type

Par défaut , aucune opération n'est "cascadée"

```java
@Entity
@Table(name="voiture")
public class Voiture {

    ...

    @OneToOne(cascade=CascadeType.ALL)
    @JoinColumn(name="moteur_id")
    private Moteur moteur;


    //getters & setters
    . . .
}
```

```java
@Entity
@Table(name="voiture")
public class Voiture {

    ...

    @OneToOne(cascade= {CascadeType.DETACH,
                        CascadeType.MERGE,
                        CascadeType.PERSIST,
                        CascadeType.REFRESH,
                        CascadeType.REMOVE   })
    @JoinColumn(name="moteur_id")
    private Moteur moteur;


    //getters & setters
    . . .
}
```