FIAP GRADUAÇÃO

FIAP

ENTERPRISE APPLICATION DEVELOPMENT

Prof. Me. Thiago T. I. Yamamoto

#03 – JPA API

 *thiagoyama@gmail.com*  *thiagoyama*

TRAJETÓRIA

- JPA Introdução
- JPA API

#03 - AGENDA

- Entity Manager
- Persistence Unit e Persistence Context
- Obter um Entity Manager
- Controlar as transações
- Estados de uma Entidade
- Métodos do Entity Manager:
  - Persist
  - Find
  - Merge
  - Refresh
  - Remove
- Métodos do ciclo de vida da Entidade

JPA – JAVA PERSISTENCE API

FIAP

- **Especificação** independente de fabricante;
- Elementos principais:
  - **Anotações** para mapeamento O/R;
  - **API** para **persistência** de entidades;
  - Linguagem de consultas **JPQL**;
- Implementação Hibernate:
  - Hibernate Core;
  - Hibernate Annotations;
  - Hibernate Entity Manager;

ENTITY MANAGER

ENTITY MANAGER

FIAP

- Unidade central para **gerenciamento** de **entidades** na JPA por meio de uma **API padronizada**;
- **Responsável** pela **criação**, **atualização**, **remoção** e **consultas** às entidades;
- Outras atribuições: controle de transações, gerenciamento de *cache*, etc...;

ENTITY MANAGER

- Uma aplicação deve ter acesso a uma instância do **Entity Manager** para realizar as operações de persistência;

Existem **duas** formas de obter-se um **Entity Manager**:

1. ***Application Managed Entity Manager:*** A **própria aplicação** obtém um Entity Manager e o fecha utilizando métodos da API apropriados;

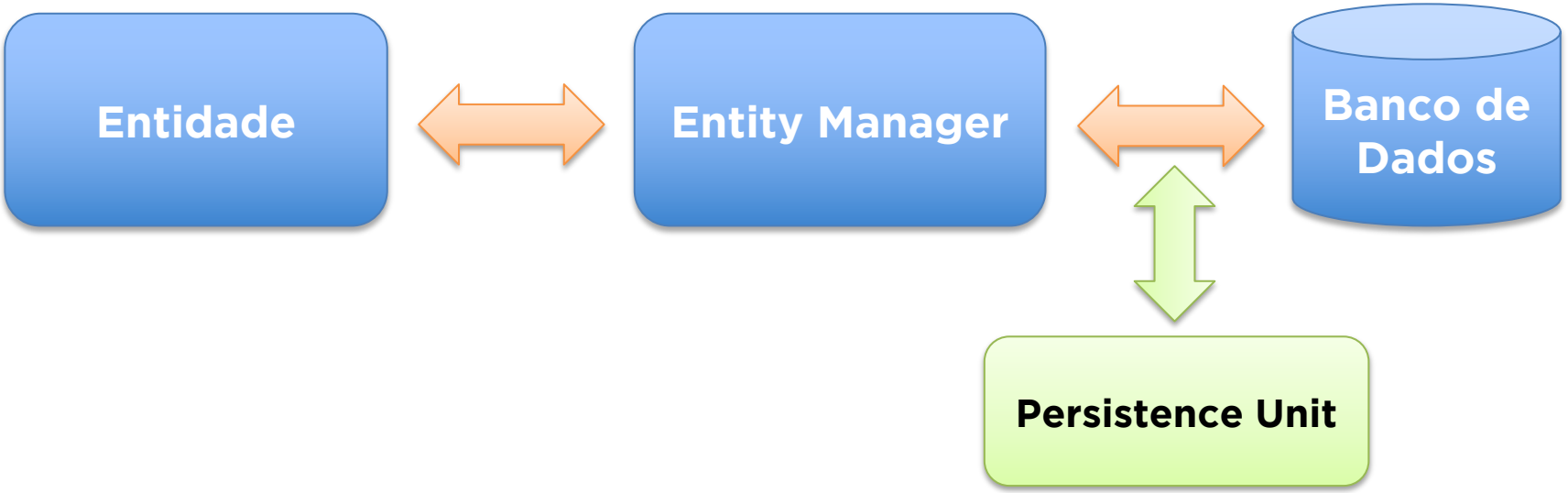
2. ***Container Managed Entity Manager:*** Um **container** instância e **gerencia** o **Entity Manager**;

Vamos trabalhar neste momento com o **Application Managed**.

PERSISTENCE UNIT

FIAP

- Uma **Persistence Unit** define todas as configurações necessárias para que um **Entity Manager** consiga efetuar a persistência de um conjunto de entidades;
- Na **Persistence Unit** definimos as configurações do banco de dados, a **URL** de conexão, usuário, senha e etc..;
- Cada **Persistence Unit** é identificada por um nome dentro do arquivo **persistence.xml** na pasta **META-INF;**



PERSISTENCE CONTEXT

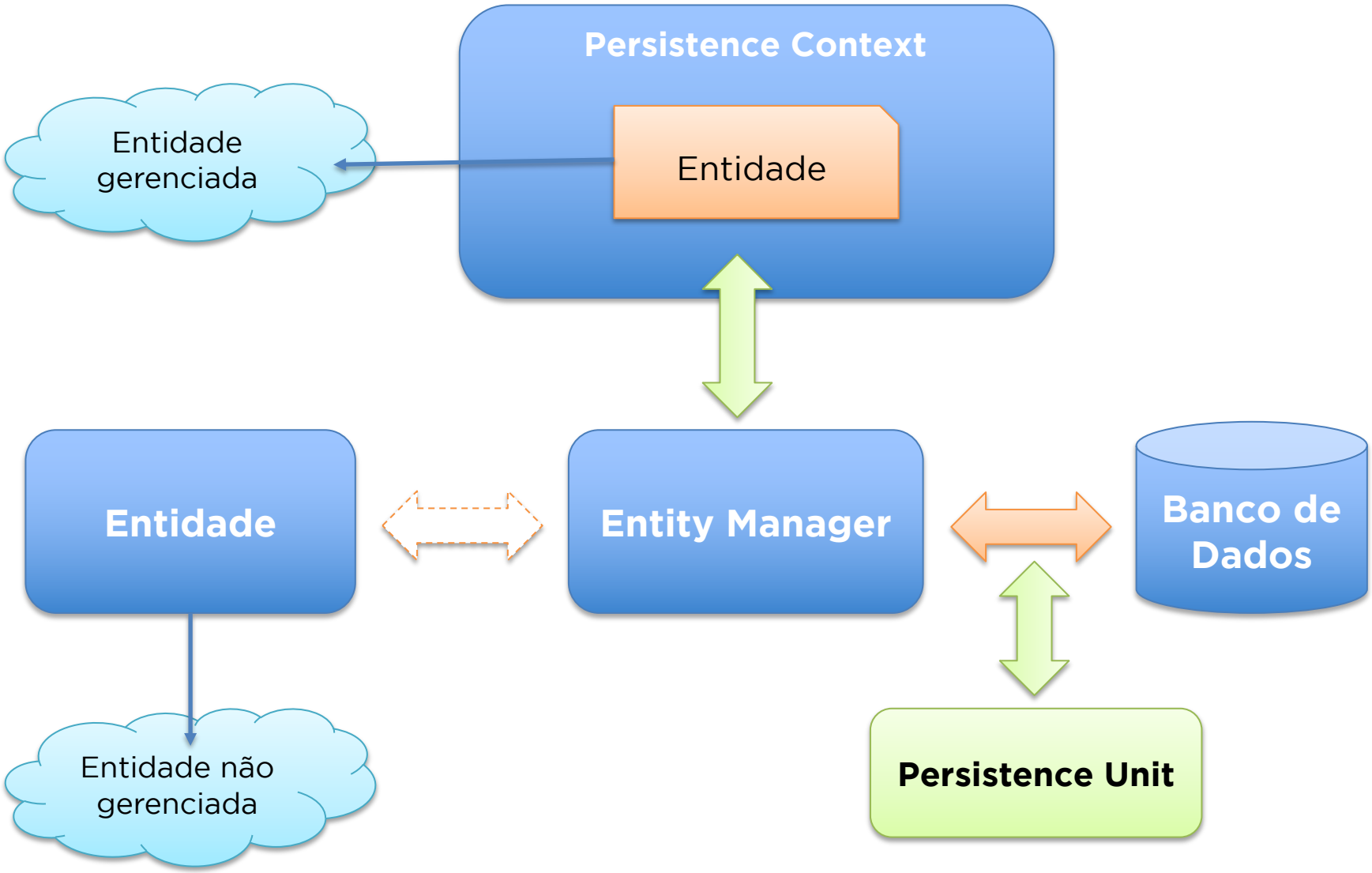
FIAP

- É o **conjunto de instâncias de entidades** “visíveis” ao **Entity Manager**, isto é, que ele pode gerenciar;
- Cada **Entity Manager** mantém apenas um único **Persistence Context**;
- Ao **fechar** um **contexto de persistência**, todas suas instâncias de entidades associadas tornam-se **não gerenciadas;**

- **Entidades Gerenciadas:**
  - Quando entidades estão associadas a um contexto de persistência;
  - Alterações no estado das entidades são sincronizadas com o banco de dados;

- **Entidades Não Gerenciadas:**
  - Entidades não associadas a um contexto de persistência;
  - Alterações nas entidades não se refletem no banco de dados;

JPA – JAVA PERSISTENCE API



INSTANCIA DE ENTITY MANAGER

FIAP

- Para obter-se um Entity Manager pela aplicação é necessário primeiro uma referência a um **EntityManagerFactory**:

```
EntityManagerFactory f = Persistence.createEntityManagerFactory("cliente");
```

  - Onde o **parâmetro**, no exemplo "**cliente**", corresponde ao **nome** de uma *persistence unit* definida no arquivo ***persistence.xml***

- Depois é só obter o **Entity Manager** a partir do **EntityManagerFactory**:

```
EntityManager em = f.createEntityManager();
```

- Para **fechar** o **Entity Manager** basta utilizar o método close:

```
em.close();
```

TRANSAÇÕES

FIAP

- As **transações** definem quanto as **entidades** devem ser **sincronizadas** com a **base de dados**;
- Quando uma transação é confirmada (**commit**) todas as alterações realizadas nas entidades são sincronizadas com a base de dados ou, pelo contrário, podem ser descartadas (**rollback**);
- O **Entity Manager** só pode ter uma **única transação ativa** por vez;

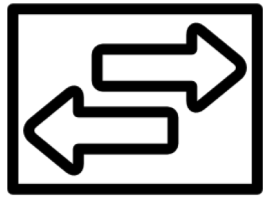
- Existem dois tipos de abordagem suportadas pela **JPA** (definidos no **persistence.xml** para cada *persistence unit*):
  - **Resource Local**: transação nativa JDBC (mais simples, de responsabilidade da aplicação);
  - **Java Transaction API (JTA)**: mecanismo padrão do Java EE (mais elaborado, o container Java EE gerencia, é transparente para a aplicação);

TRANSAÇÕES

- Para obter-se uma transação do **Entity Manager** utilizar o método **getTransaction();**
- Uma transação é representada pela classe **EntityTransaction** que contém os métodos principais:
  - **begin()**: delimita o início de uma transação (é obrigatório, um por vez);
  - **commit()**: confirma a transação (sincroniza o contexto de persistência com a base de dados);
  - **rollback()**: desfaz a transação;
  - **isActive()**: verifica se a transação encontra-se em andamento, isto é, não ocorreu *commit* ou *rollback* ainda;

Exemplo

```
EntityManager em = ...;
EntityTransaction t = em.getTransaction();
t.begin();
// métodos da JPA
t.commit(); // ou t.rollback();
```

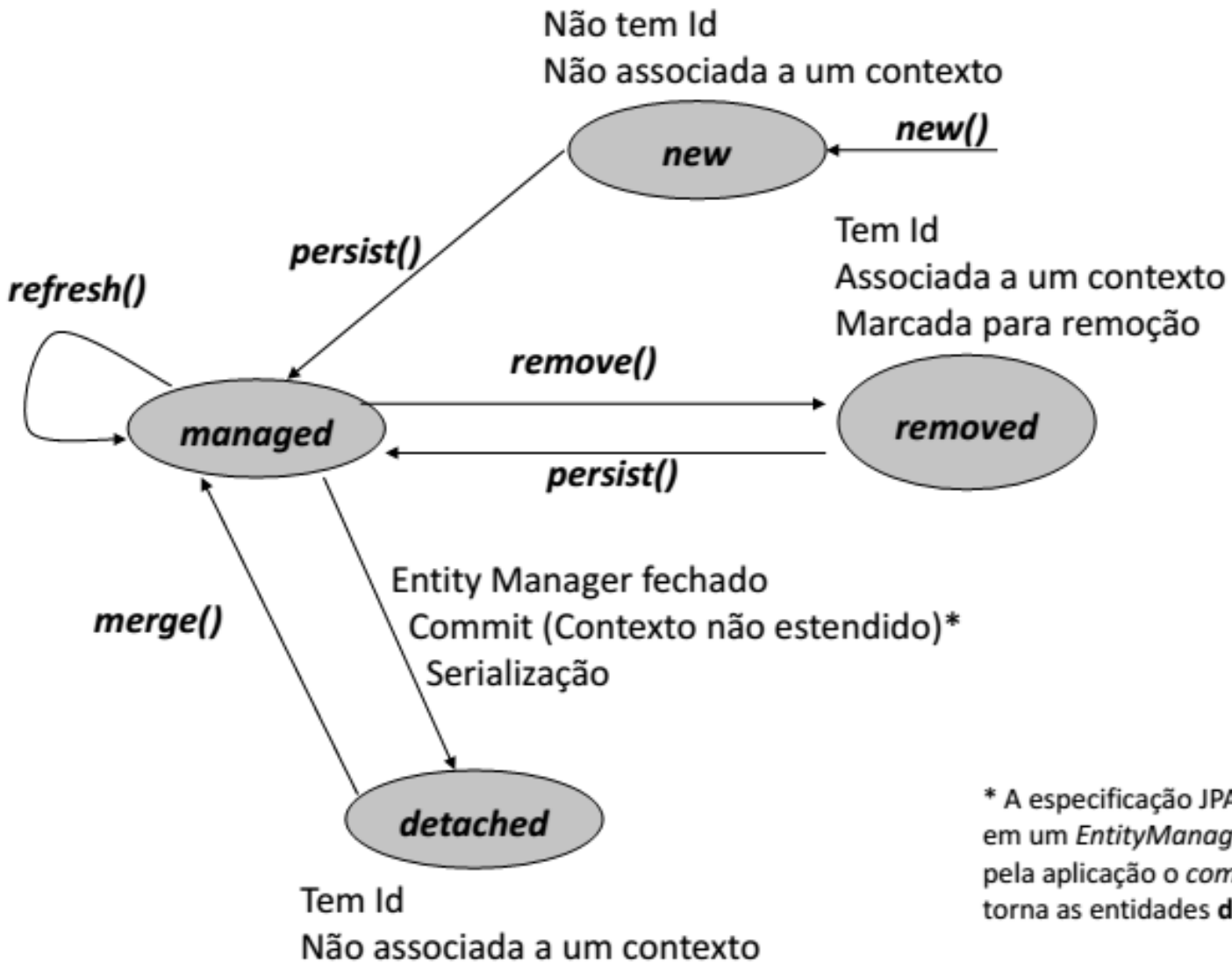


ESTADOS DA ENTIDADE

Uma entidade pode encontrar-se em um dos quatro estados abaixo:

- **new**: **instância da entidade criada** em memória mas nunca foi associada a um contexto de persistência e **não possui id** equivalente no banco de dados;
- **managed**: **tem um id no banco de dados** e está atualmente associada a um contexto de persistência;
- **detached**: **tem um id no bando de dados** mas **não** está atualmente **associada ao contexto de persistência;**
- **removed**: instância da entidade associada a um contexto de persistência mas está **programada para ser removida** do banco de dados;

ESTADOS DA ENTIDADE



* A especificação JPA define que em um *EntityManager* gerenciado pela aplicação o *commit* não torna as entidades **detached**.

MÉTODOS DO ENTITY MANAGER

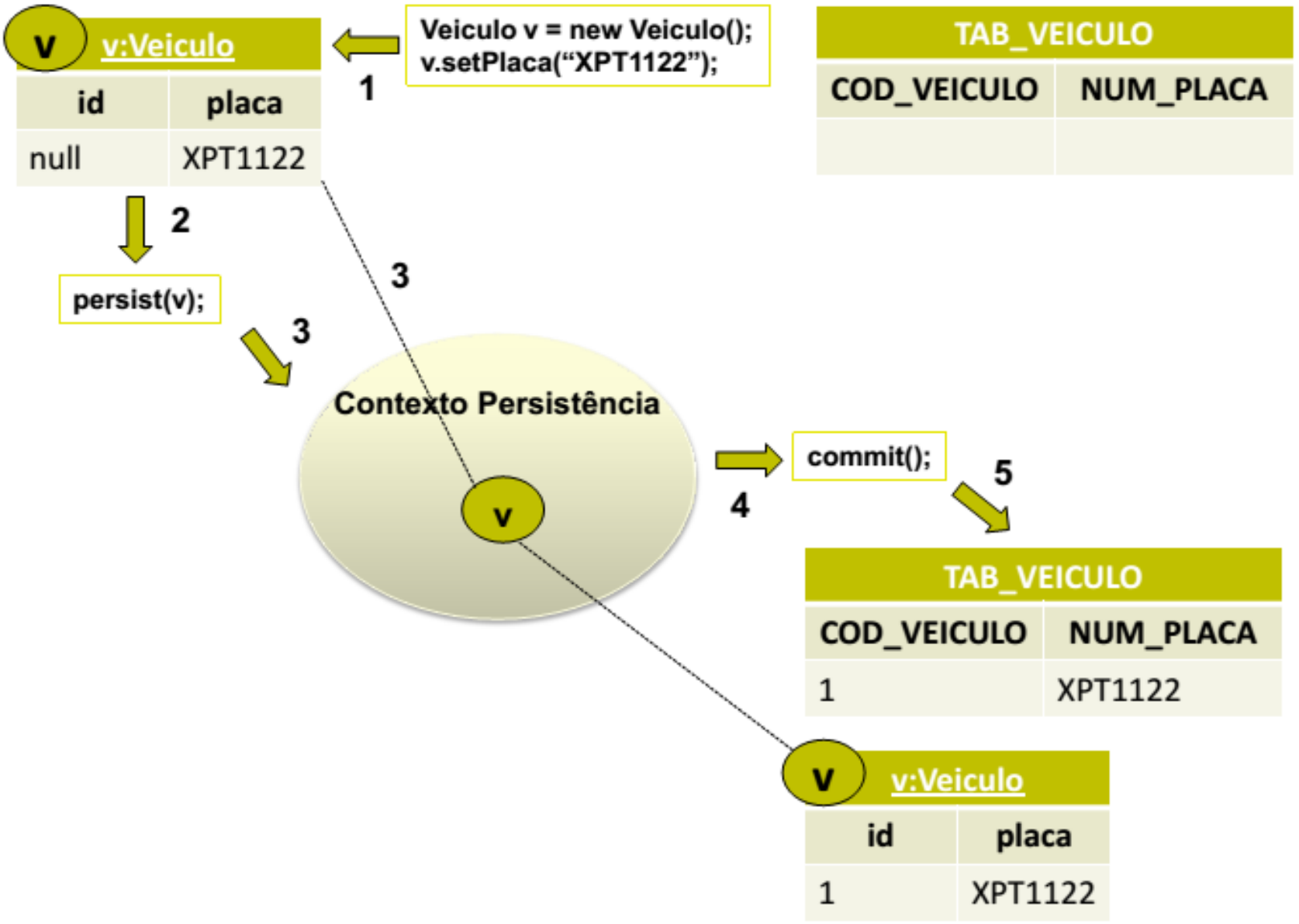
PERSIST

- **persist (Object entity)**: enfileira uma nova entidade para ser inserida no banco de dados e a torna gerenciada;
  - ✓ Caso a entidade seja **NEW** então torna-se **MANAGED;**
  - ✓ Caso a entidade seja **MANAGED** ela é ignorada;
  - ✓ Caso a entidade seja **REMOVED** então torna-se **MANAGED;**
  - ✓ Caso a entidade seja **DETACHED**, uma **IllegalArgumentException** é lançada;

```
Veiculo veiculo = new Veiculo();
veiculo.setPlaca("DHZ-5678");
veiculo.setModelo(”Gol");

manager.persist(veiculo);
```

PERSIST

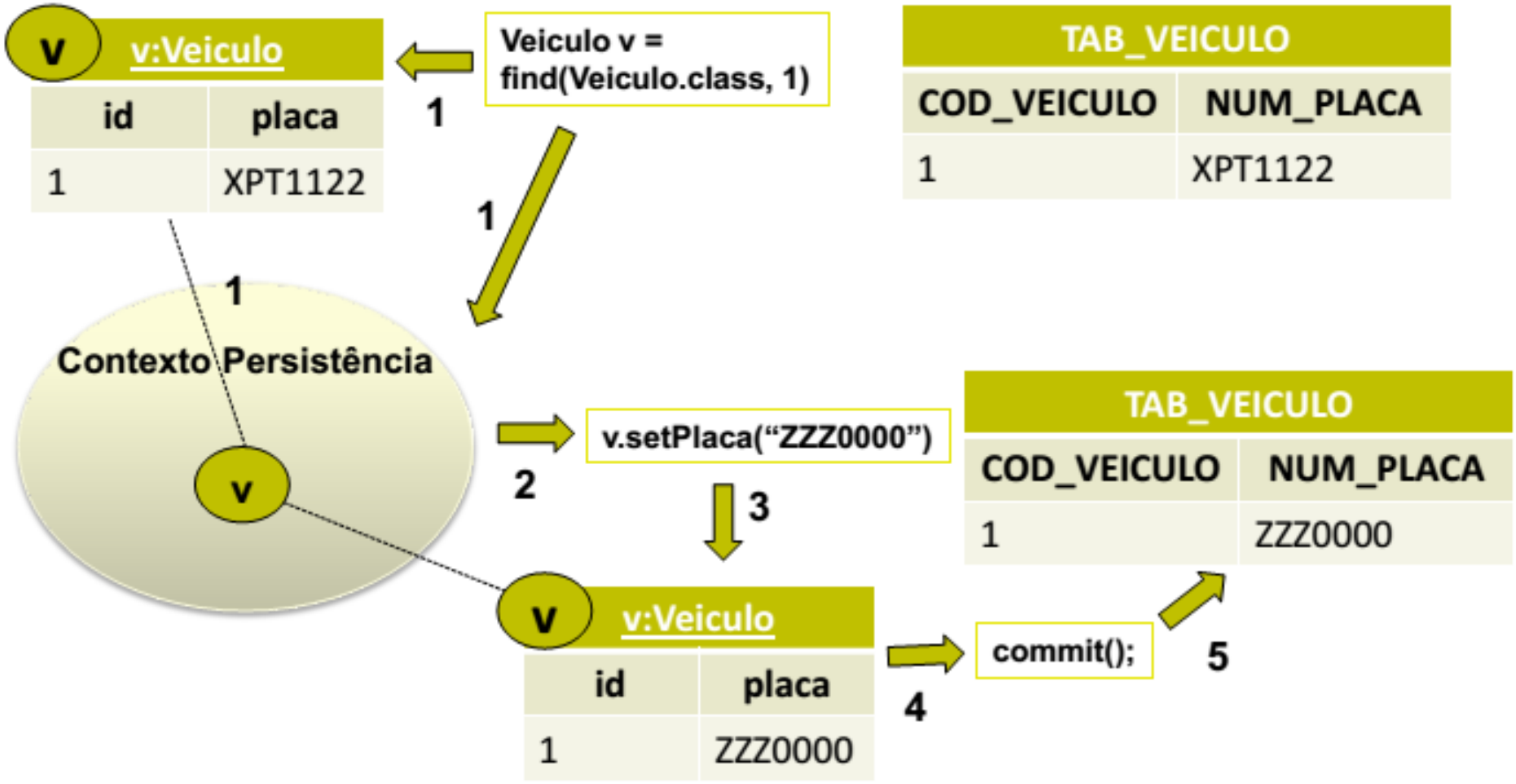

v
v:Veiculo
id
placa
null
XPT1122
1
Veiculo v = new Veiculo();
v.setPlaca("XPT1122");
2
persist(v);
3
3
Contexto Persistência
v
4
commit();
5
TAB_VEICULO
COD_VEICULO
NUM_PLACA
TAB_VEICULO
COD_VEICULO
NUM_PLACA
1
XPT1122
v
v:Veiculo
id
placa
1
XPT1122

FIND

- **<T> T find (Class<T> classeEntidade, Object PK)**: realiza uma busca por meio da chave primária da entidade;
  - ✓ Retorna uma instância **MANAGED** caso seja localizada ou *null* caso contrário;

```
// Busca veiculo com id igual a 10
Veiculo veiculo = manager.find(Veiculo.class, 10);
```

FIND


v
v:Veiculo
id
placa
1
XPT1122
1
Veiculo v = find(Veiculo.class, 1)
1
1
Contexto Persistência
v
2
v.setPlaca("ZZZ0000")
3
v
v:Veiculo
id
placa
1
ZZZ0000
4
commit();
5
TAB_VEICULO
COD_VEICULO
NUM_PLACA
1
XPT1122
TAB_VEICULO
COD_VEICULO
NUM_PLACA
1
ZZZ0000

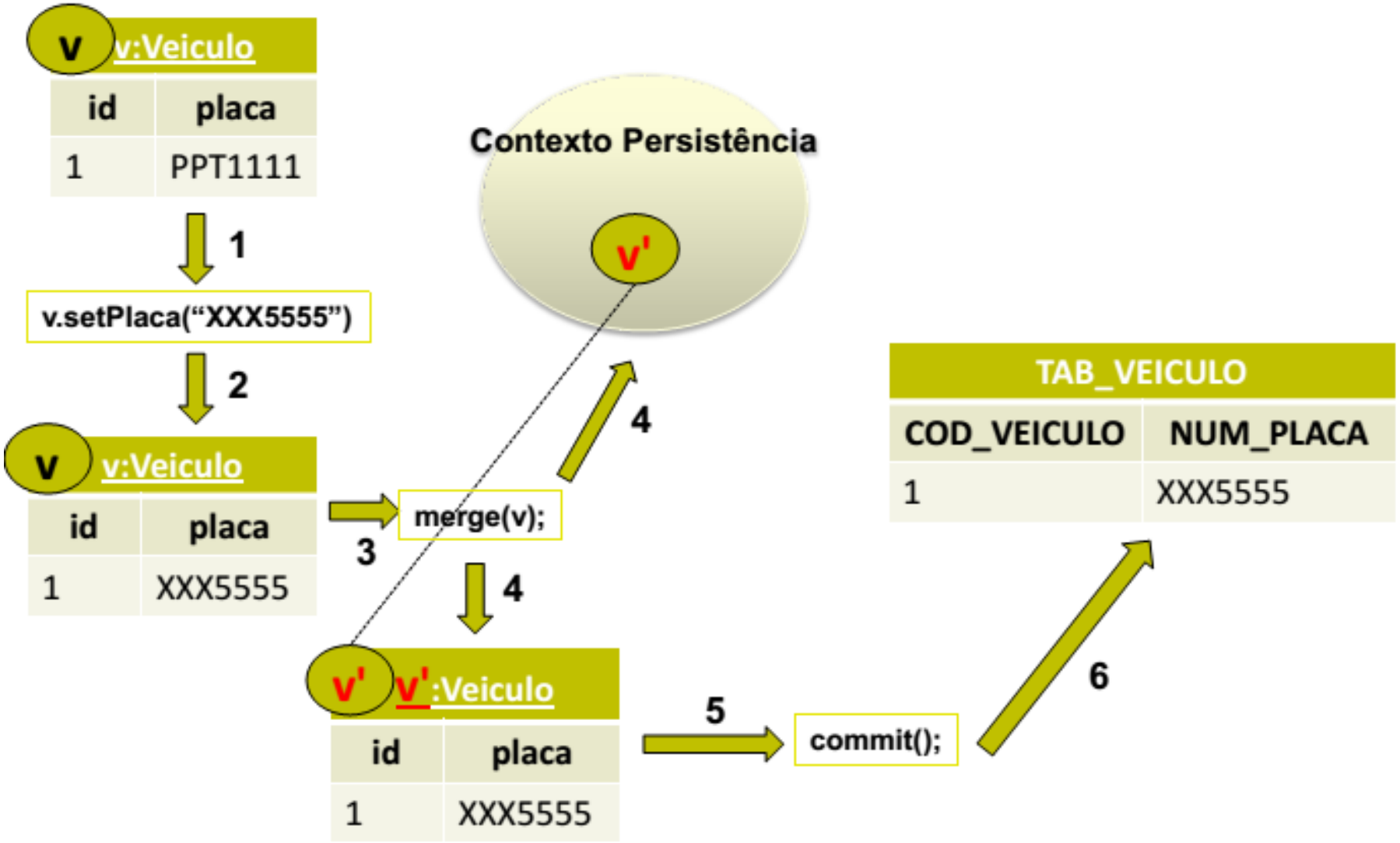
MERGE

FIAP

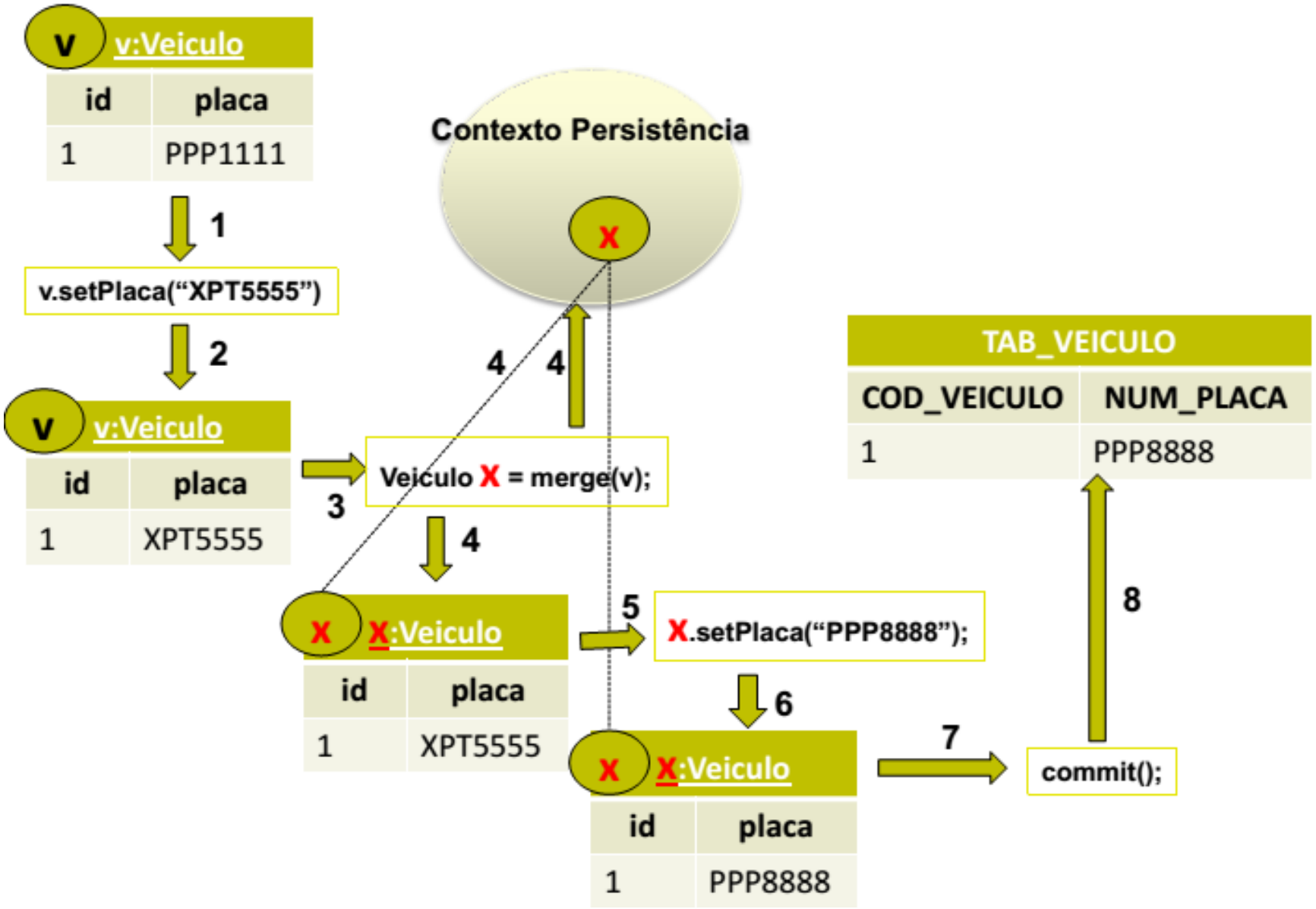
- **<T> T merge (T entidade)**: retorna uma cópia gerenciada de uma entidade não gerenciada;
  - ✓ Caso a entidade seja **DETACHED**, seu estado é copiado para uma instância **MANAGED** com a mesma identidade ou uma nova cópia **MANAGED** da entidade é criada;
  - ✓ Caso a entidade seja **NEW**, uma nova entidade **MANAGED** é criada com o estado copiado da entidade original;
  - ✓ Caso a entidade seja **MANAGED** ela é ignorada;
  - ✓ Caso a entidade seja **REMOVED**, uma **IllegalArgumentException** é lançada;

```
Veiculo v2 = manager.merge(veiculo);
v2.setPlaca("DHZ-5678");
v2.setModelo("Fusca");
```

MERGE


v
v:Veiculo
id
placa
1
PPT1111
1
v.setPlaca("XXX5555")
2
v
v:Veiculo
id
placa
1
XXX5555
3
merge(v);
4
Contexto Persistência
v'
4
v'
v':Veiculo
id
placa
1
XXX5555
5
commit();
6
TAB_VEICULO
COD_VEICULO
NUM_PLACA
1
XXX5555

MERGE

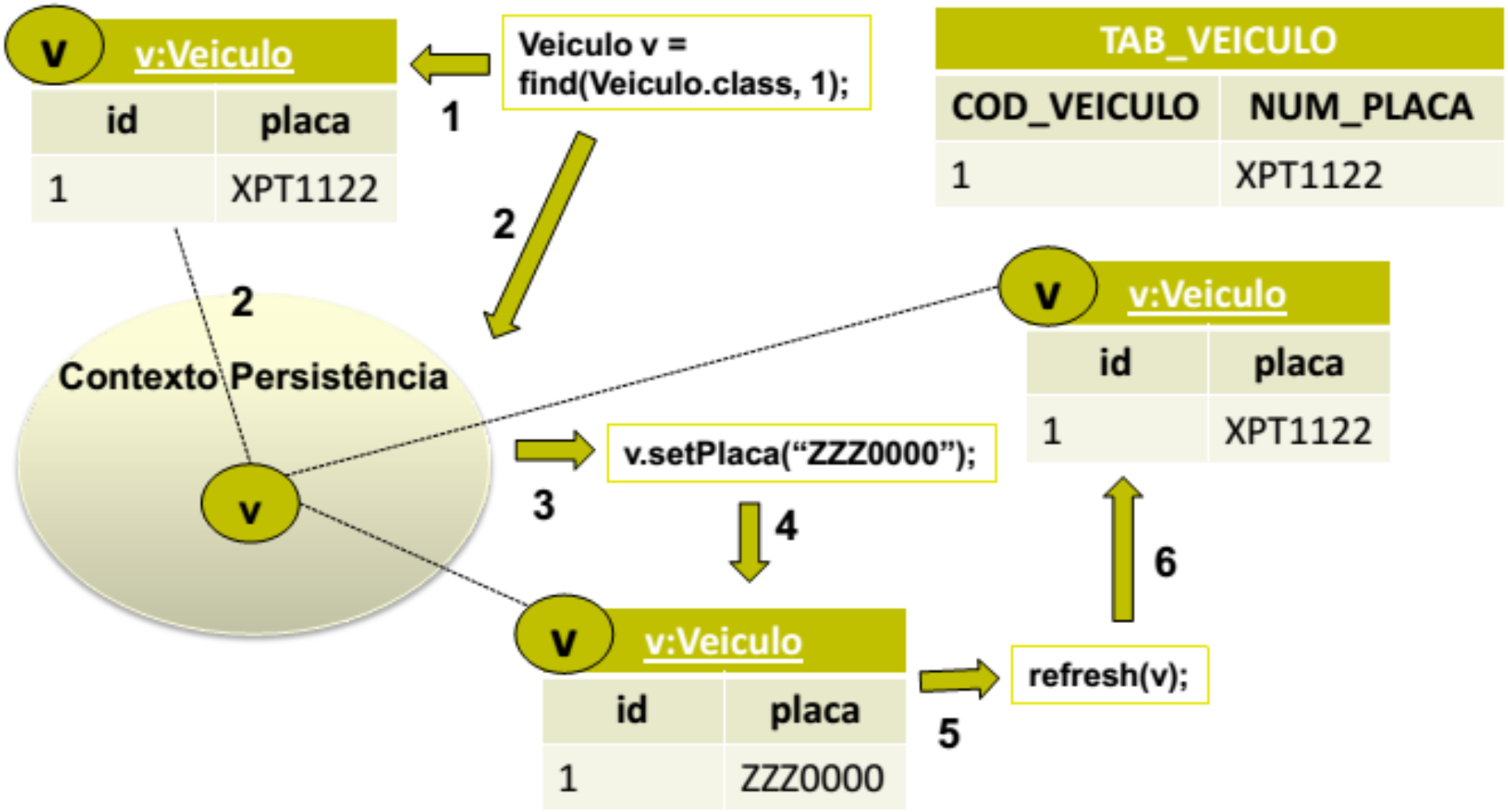

v
v:Veiculo
id
placa
1
PPP1111
1
v.setPlaca("XPT5555")
2
v
v:Veiculo
id
placa
1
XPT5555
3
Veiculo X = merge(v);
4
X
X:Veiculo
id
placa
1
XPT5555
Contexto Persistência
X
4
4
5
X.setPlaca("PPP8888");
6
X
X:Veiculo
id
placa
1
PPP8888
7
commit();
8
TAB_VEICULO
COD_VEICULO
NUM_PLACA
1
PPP8888

REFRESH

- **refresh(Object entidade)**: certifica que o estado da entidade encontra-se sincronizado com a base de dados;
  - ✓ Caso a entidade seja **NEW** ela é ignorada;
  - ✓ Caso a entidade seja **MANAGED**, seu estado é atualizado com a base de dados;
  - ✓ Caso a entidade seja **REMOVED**, ela é ignorada;
  - ✓ Caso a entidade seja **DETACHED**, uma **IllegalArgumentException** é lançada;

```
Veiculo veiculo = new Veiculo();
veiculo = manager.find(Veiculo.class, 1);
veiculo.setPlaca("DHZ-5678");
veiculo.setModelo("Fusca");
manager.refresh(veiculo);
```

REFRESH


v
v:Veiculo
id
placa
1
XPT1122
Veiculo v =
find(Veiculo.class, 1);
1
2
2
Contexto Persistência
v
TAB_VEICULO
COD_VEICULO
NUM_PLACA
1
XPT1122
v
v:Veiculo
id
placa
1
XPT1122
v.setPlaca("ZZZ0000");
3
4
6
v
v:Veiculo
id
placa
1
ZZZ0000
refresh(v);
5

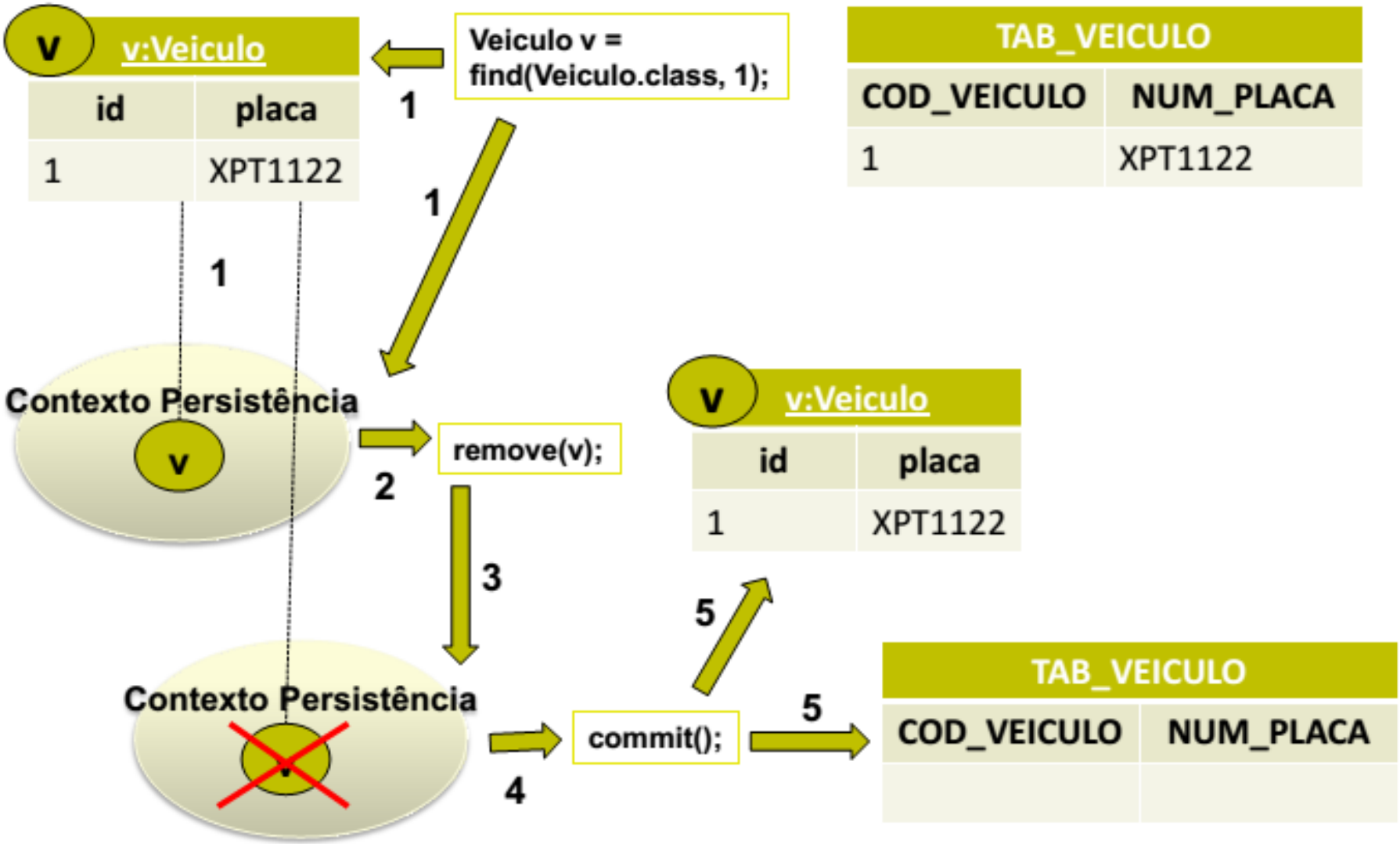
REMOVE

- **remove(Object entidade)**: marca uma entidade como **REMOVED** que será excluída do banco de dados após uma instrução **commit;**
  - ✓ Caso a entidade seja **NEW**, ela é ignorada;
  - ✓ Caso a entidade seja **MANAGED**, ela torna-se **REMOVED;**
  - ✓ Caso a entidade seja **REMOVED** ela é ignorada;
  - ✓ Caso a entidade seja **DETACHED**, uma **IllegalArgumentException** é lançada;

Para excluir entidades desacopladas primeiro deve-se torná-la **MANAGED**, por exemplo, utilizando o método **find;**

```
Veiculo veiculo = manager.find(Veiculo.class, 10);
manager.remove(veiculo);
```

REMOVE


v
v:Veiculo
id
placa
1
XPT1122
Veiculo v =
find(Veiculo.class, 1);
1
1
1
Contexto Persistência
v
remove(v);
2
3
Contexto Persistência
4
commit();
5
5
v
v:Veiculo
id
placa
1
XPT1122
TAB_VEICULO
COD_VEICULO
NUM_PLACA
1
XPT1122
TAB_VEICULO
COD_VEICULO
NUM_PLACA

MÉTODOS DO CICLO DE VIDA

- Pode-se realizar ações em cada fase do ciclo de vida de uma entidade;
- Para tanto, basta utilizar as anotações abaixo:
  - ✓ **@PrePersist**
  - ✓ **@PostPersist**
  - ✓ **@PreRemove**
  - ✓ **@PostRemove**
  - ✓ **@PreUpdate**
  - ✓ **@PosUpdate**
  - ✓ **@PostLoad**

VOCÊ APRENDEU...

- O que é o **Entity Manager**, **Persistence Unit** e **Persistence Context**;
- Obter uma **instância** do Entity Manager e a controlar as **transações**;
- Sobre os **estados** da **entidade** e os **métodos:**
  - Persist;
  - Merge;
  - Find;
  - Refresh;
  - Remove;
- Métodos do **ciclo de vida** da entidade;

FIAP

"Aprender é a única coisa que a mente nunca se cansa, nunca tem medo e nunca se arrepende"