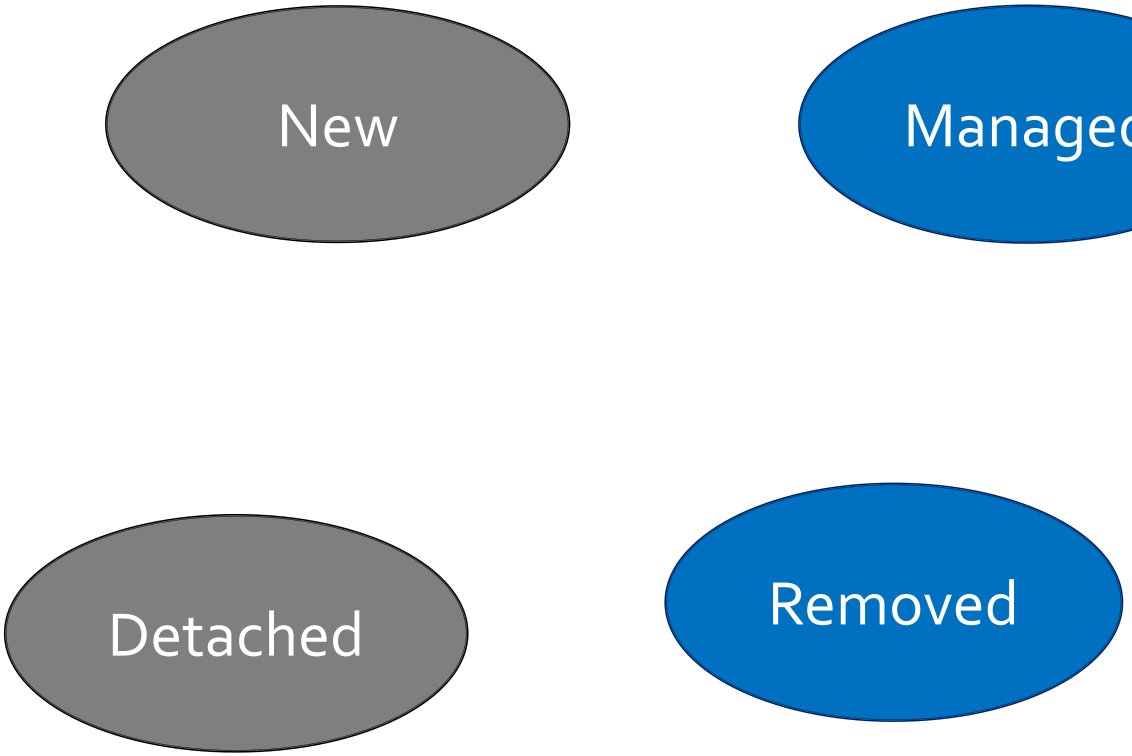
EAPLI : Engenharia de Aplicações

ORM

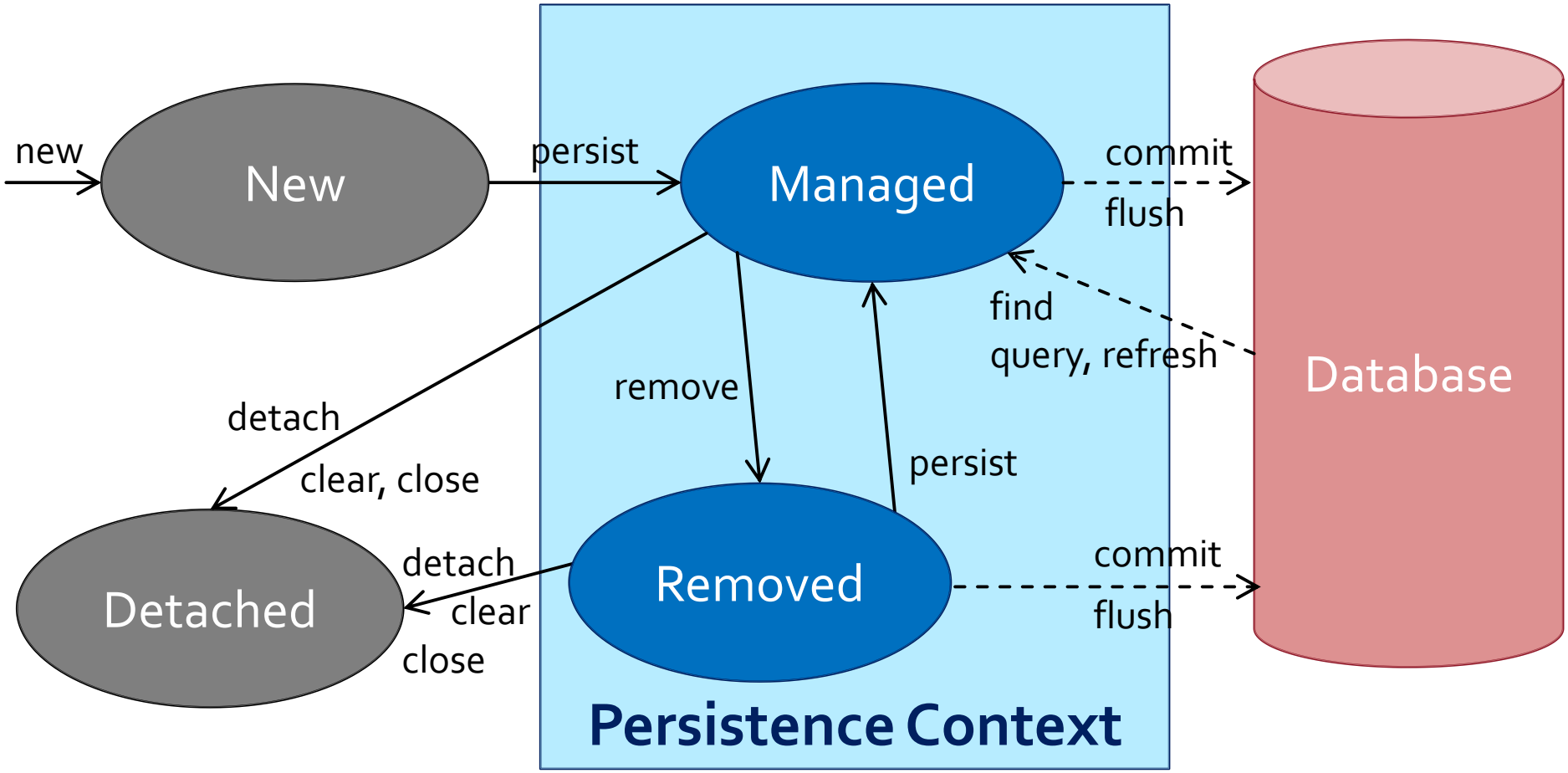
. Entity Lifecycle

Entity Lifecycle

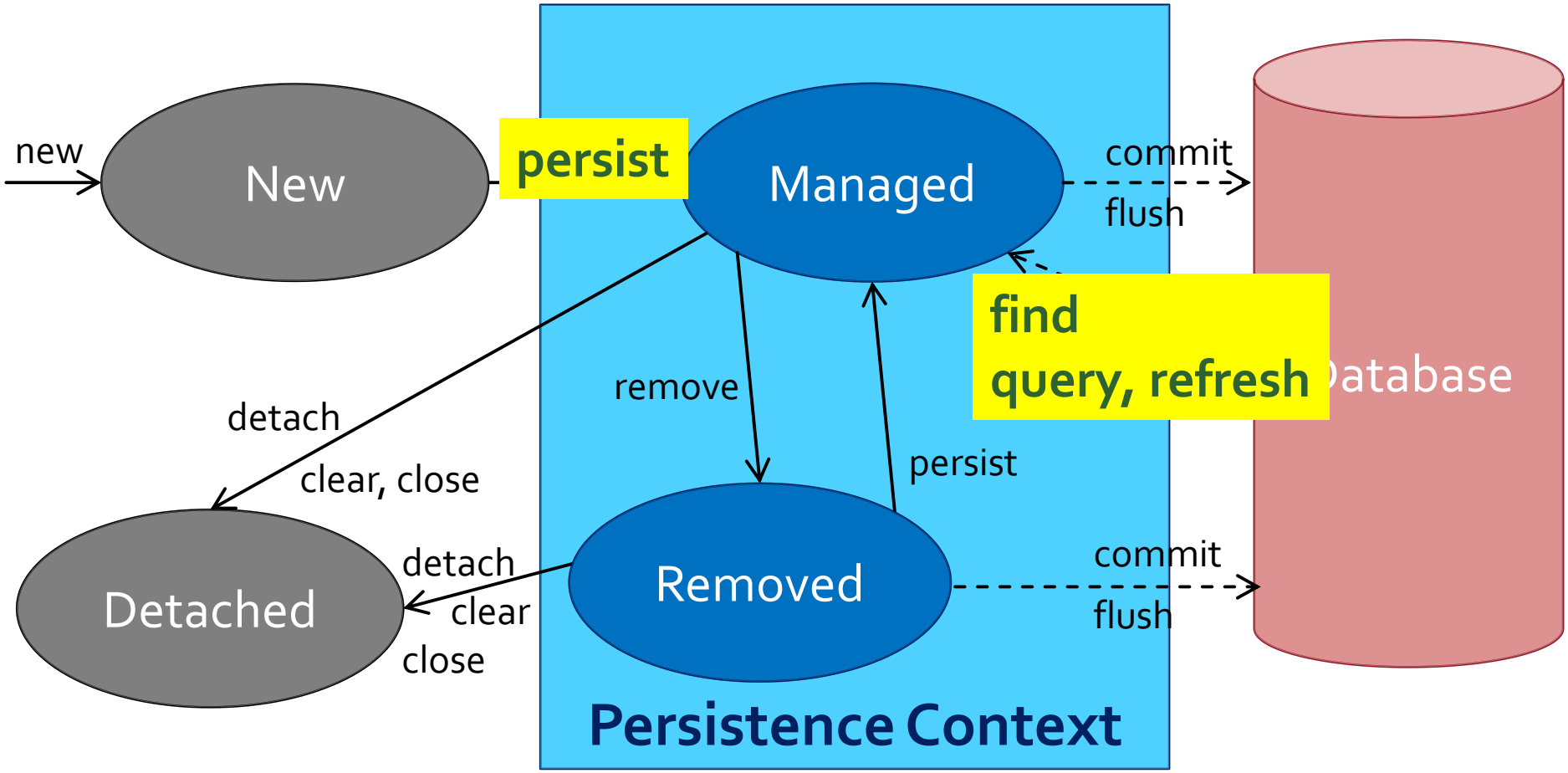
Ciclo de vida das Entidades



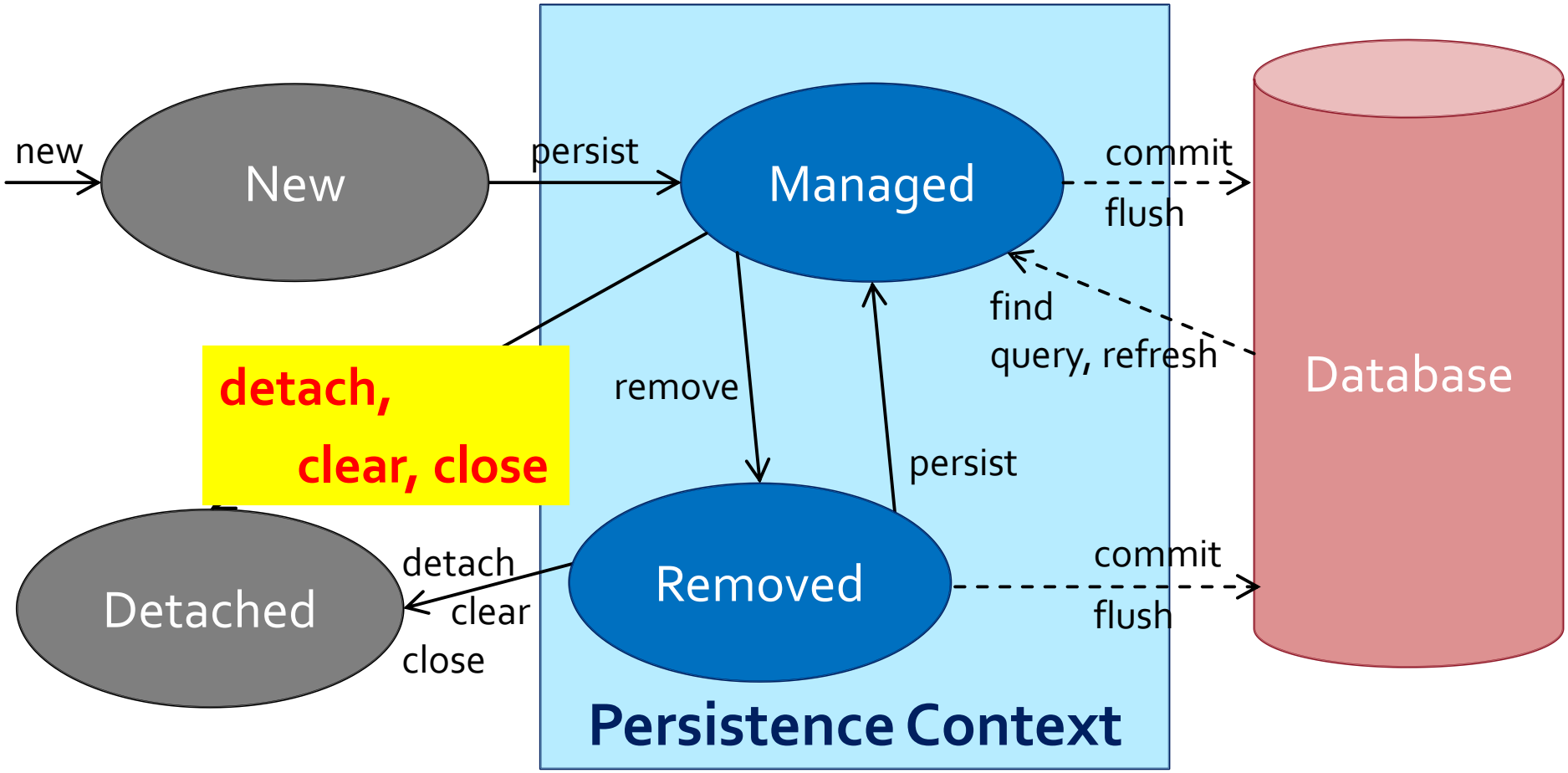
Ciclo de vida das Entidades



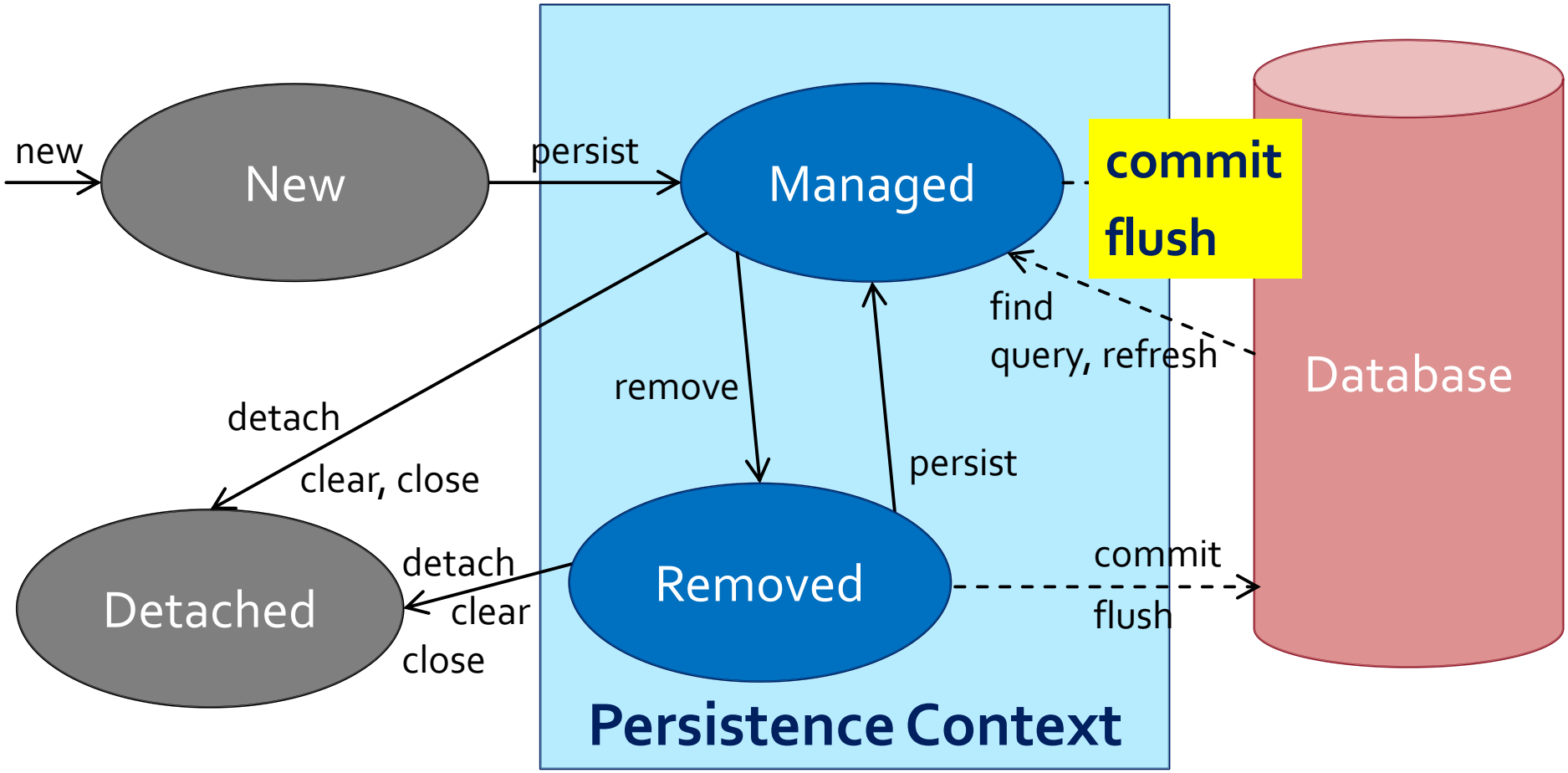
Ciclo de vida das Entidades



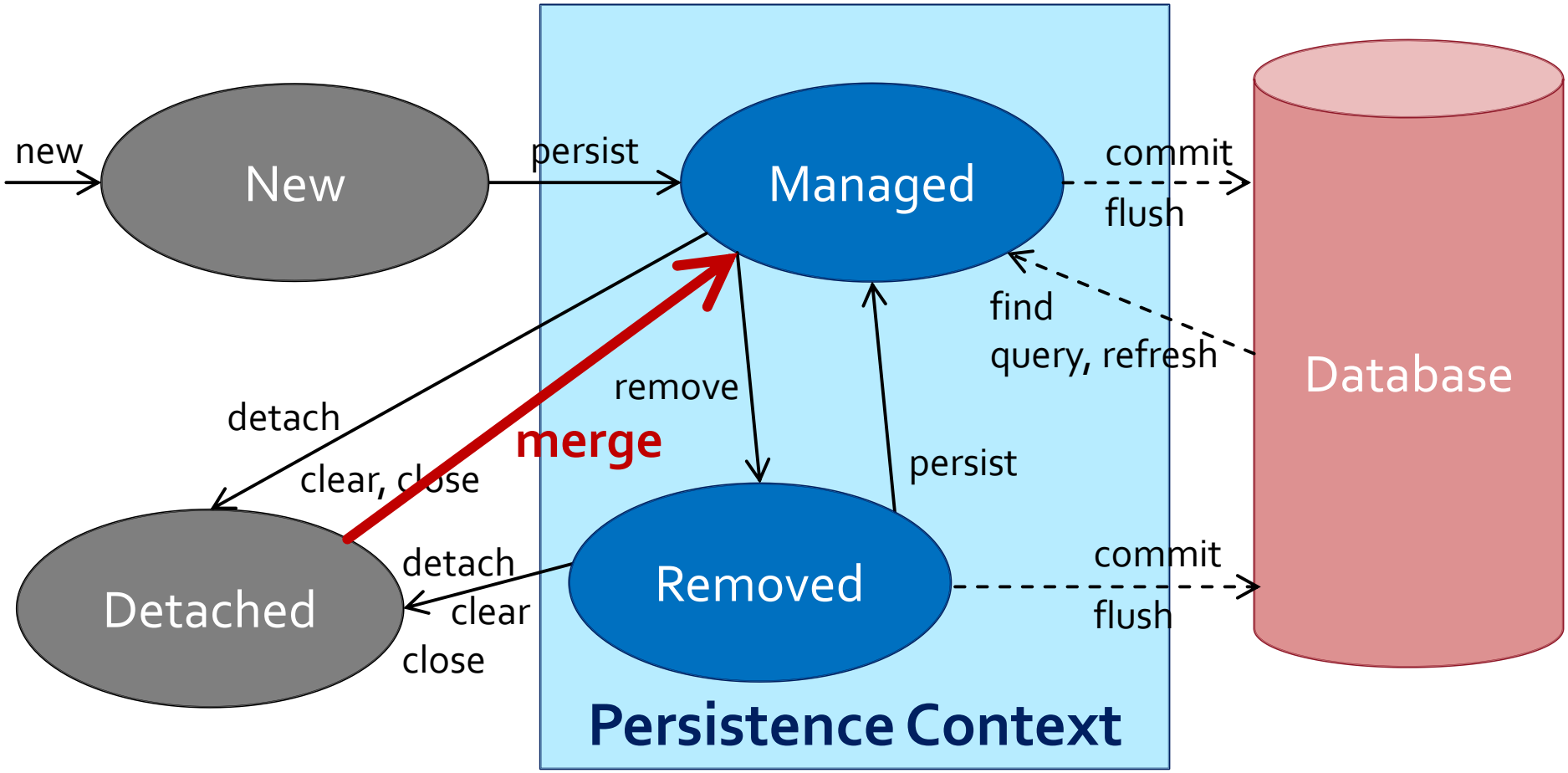
Ciclo de vida das Entidades



Ciclo de vida das Entidades



Ciclo de vida das Entidades



Ciclo de vida das Entidades

EntityManager methods to manage entities lifecycle

clear: Clear the persistence context, causing all managed entities to become detached

close:Close an entity manager

commit:Commit the current transaction *(EntityTransaction an interface used to control transactions on resource-local entity managers: begin, commit, rollback, isActive)*

contains:Check if the instance is a managed entity instance belonging to the current persistence context

detach: Remove the given entity from the persistence context, causing a managed entity to become detached

find: Find by primary key, creates a new instance loading its status from the database

flush: Synchronize the persistence context to the underlying database

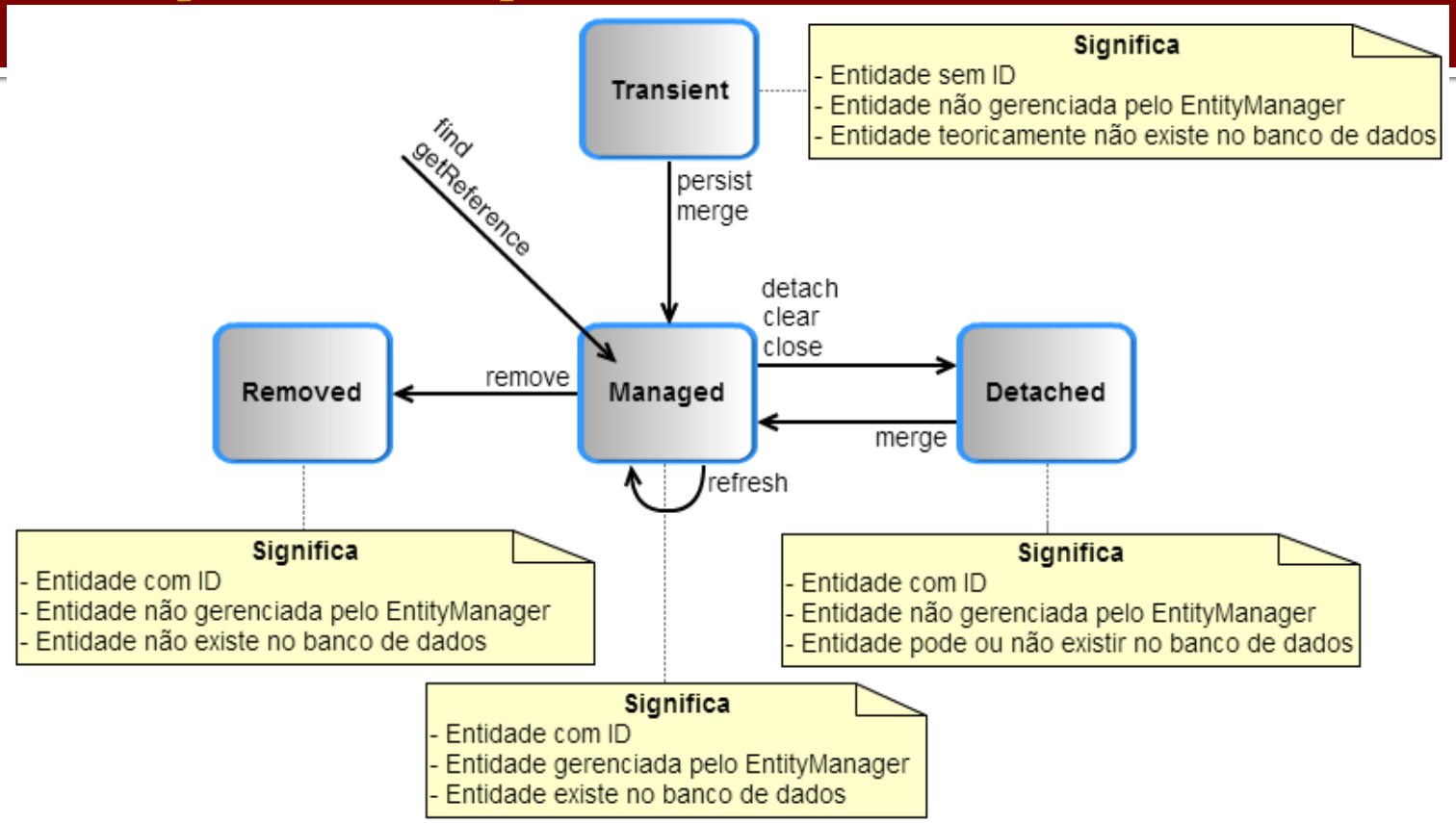
merge: Merge the state of the given entity into the current persistence context

persist: Make an instance managed and persistent

refresh: Refresh the state of the instance from the database, overwriting changes made to the entity, if any

remove: Remove the entity instance

Entity Lifecycle



O Persistence Context é a coleção de entidades geridas pelo Entity Manager. Ao carregar um objeto que já exista no Persistence Context, o objeto existente é retornado sem aceder à base de dados, exceto se é pedido pelo método **refresh**, o qual acede sempre à base de dados.
No entanto, cada instância do EntityManager garante que o seu Persistence Context gere uma única instância correspondente a um registo da base de dados.

Entity Lifecycle

Let us consider the following entity:

```
@Entity
public class Pessoa {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
	private String nome;
	private String morada;

	private Pessoa() {
	}

	public Pessoa(String nome, String morada) {
		this.nome = nome;
		this.morada = morada;
	}

	public int getId() {
		return id;
	}

	public void alterarMorada(String morada) {
		this.morada = morada;
	}
}
```

Entity Lifecycle

1. Estado Transient ou New

Quando um objeto entidade é criado o seu estado é **Transient** ou **New**. Neste estado o objeto existe em memória mas não está associado com um EntityManager nem tem representação na base de dados.

```
public static void main(String[] args) {
    //criar EntityManager (em) ...
    Pessoa p1 = new Pessoa("José Manuel", "Porto");
    em.getTransaction().begin();
    p1.alterarMorada("Vila Nova de Gaia");
    em.getTransaction().commit();
    System.out.println("ID gerado: " + p1.getId());
    em.close();
    emf.close();
}
```

```
SELECT ID FROM PESSOA WHERE ID <> ID
CREATE TABLE PESSOA (ID INTEGER IDENTITY NOT NULL, MORADA
VARCHAR, NOME VARCHAR, PRIMARY KEY (ID))
```

Tabela criada vazia: PESSOA(ID, MORADA , NOME)

Saída produzida pelo programa: ID gerado: null

A entidade Pessoa p1 não possui Id, nem tem representação na base de dados.

Entity Lifecycle

1. Estado Transient ou New

Quando um objeto entidade é cri... **Transient** ou **New**. Neste estado o objeto existe em memória mas não está assoc... dados.

```
public static void main(String[] a
    //criar EntityManager (em) ...
    Pessoa p1 = new Pessoa("José
    em.getTransaction().begin();
    p1.alterarMorada("Vila Nova d
    em.getTransaction().commit()
    System.out.println("ID gerado
    em.close();
    emf.close();
}
```

```
SEL
CRE
VAR
```

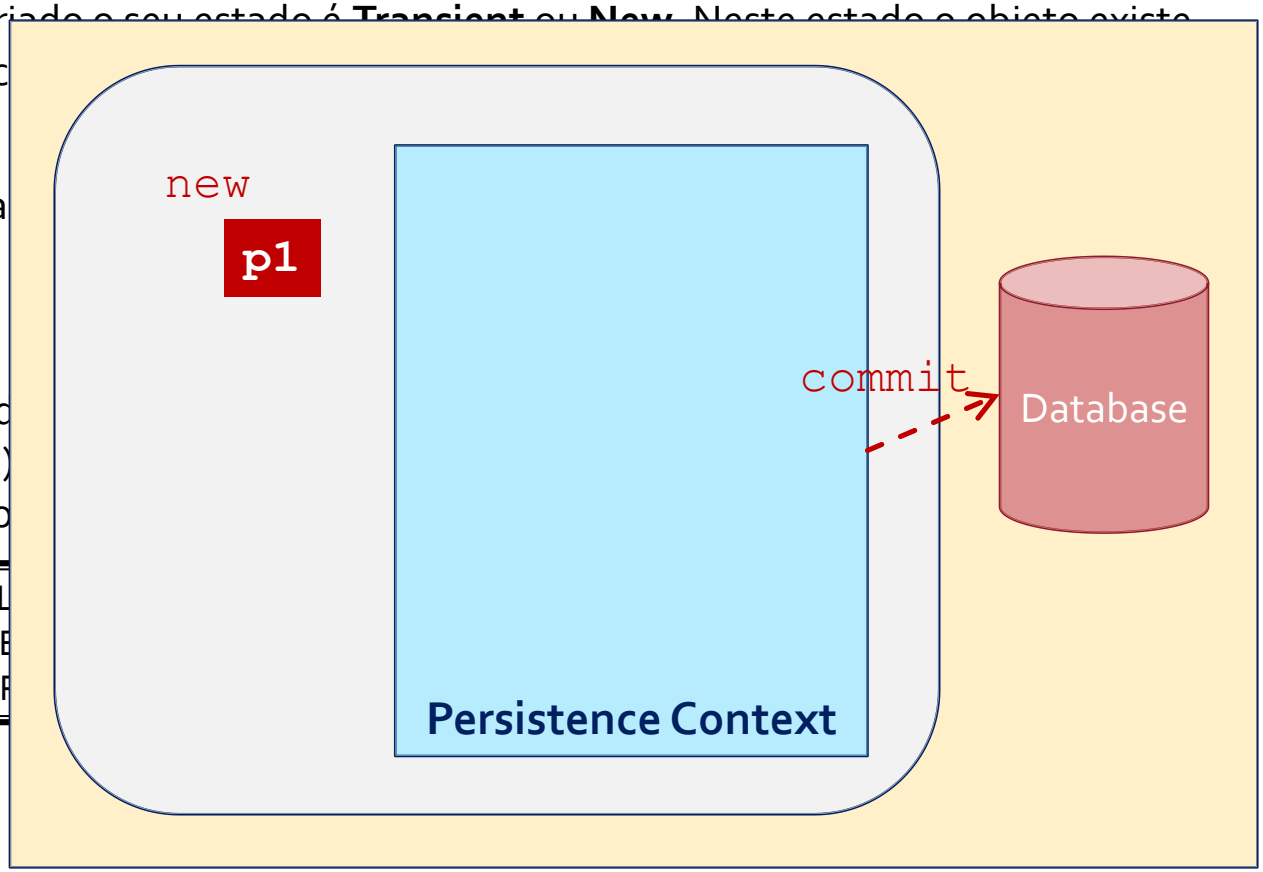


Tabela criada vazia:

Saída produzida pelo programa:

A entidade Pessoa p1 não possui Id, nem tem representação na base de dados.

Entity Lifecycle

2. Estado Managed (persist)

Um objeto entidade passa para o estado **Managed** quando é persistido na base de dados (através do método persist de um EntityManger) ou quando é carregado da base de dados (através do método find de um EntityManger, ou da execução de uma query).

```
public static void main(String[] args) {
   //criar EntityManager (em) ...
    Pessoa p1 = new Pessoa("José Manuel", "Porto");
    em.getTransaction().begin();
    em.persist(p1);
    p1.alterarMorada("Vila Nova de Gaia");
    em.getTransaction().commit();
    System.out.println("ID gerado: " + p1.getId());
    em.close();
    emf.close();
}
```

Saída produzida pelo programa: ID gerado: 1

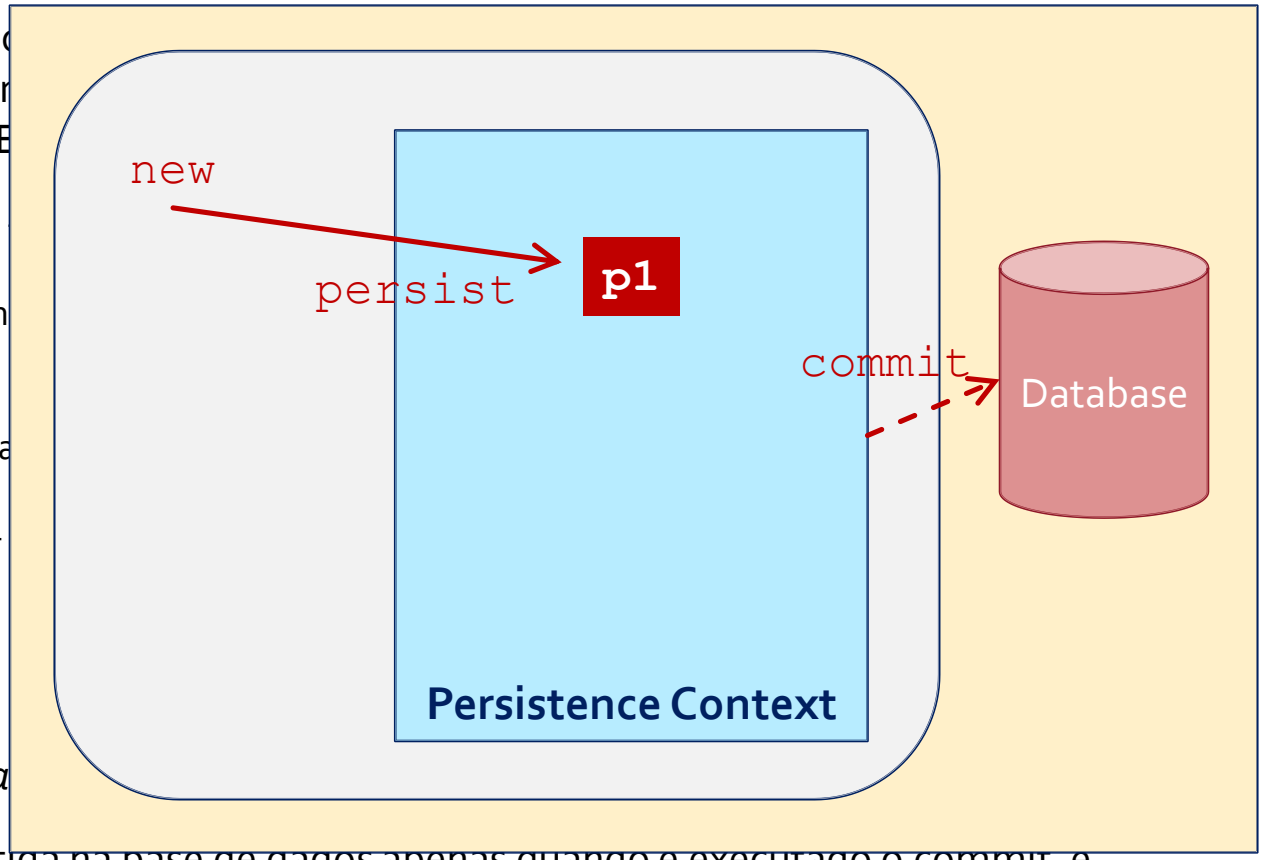
A entidade Pessoa p1 é persistida na base de dados apenas quando é executado o commit, e passa a ter ID.

Entity Lifecycle

2. Estado Managed (persist)

Um objeto entidade passa para o
(através do método persist de ur
(através do método find de um E

```
public static void main(String[] args)
   //criar EntityManager (em) ...
    Pessoa p1 = new Pessoa("José Man
    em.getTransaction().begin();
    em.persist(p1);
    p1.alterarMorada("Vila Nova de Ga
    em.getTransaction().commit();
    System.out.println("ID gerado: " +
    em.close();
    emf.close();
}
```



Saída produzida pelo programa

A entidade Pessoa p1 é persistida na base de dados apenas quando é executado o commit, e passa a ter ID.

Entity Lifecycle

3. Estado Managed (find)

No estado managed as entidades têm uma identidade persistente, uma chave que identifica univocamente cada instância. Se uma entidade managed é modificada dentro de uma transação, é marcada pelo EntityManager como dirty, e as modificações são atualizadas na base de dados no commit da transação.

```
public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPAPessoaPU");
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();
    Pessoa p1 = em.find(Pessoa.class, 1);
    p1.alterarMorada("Matosinhos");
    em.getTransaction().commit();
    System.out.println("ID gerado: " + p1.getId());
    em.close();
    emf.close();
}
```

ID	MORADA	NOME
1	Matosinhos	José Manuel

Qualquer alteração no estado do objeto é persistida na base de dados

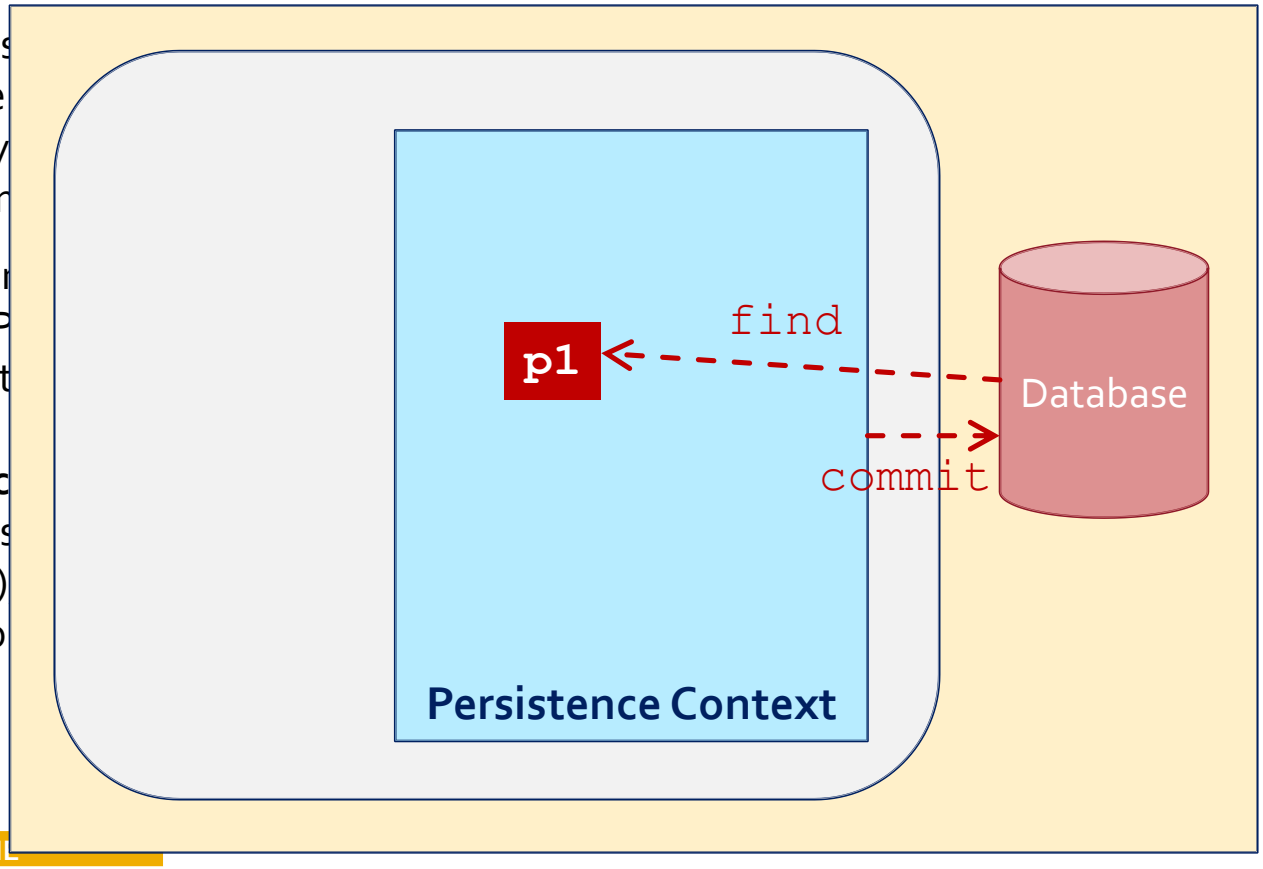
Saída produzida pelo programa: ID gerado: 1

Entity Lifecycle

3. Estado Managed (find)

No estado managed as entidades
univocamente cada instância. Se
transação, é marcada pelo Entity
base de dados no commit da tran

```
public static void main(String[] ar
    EntityManagerFactory emf = P
    EntityManager em = emf.creat
    em.getTransaction().begin();
    Pessoa p1 = em.find(Pessoa.c
    p1.alterarMorada("Matosinhos
    em.getTransaction().commit()
    System.out.println("ID gerado
    em.close();
    emf.close();
}
```



ID	MORADA	NOME
1	Matosinhos	José Manuel

Qualquer alteração no estado do objeto é persistida na base de dados

Saída produzida pelo programa: ID gerado: 1

Entity Lifecycle

4. Estado Detached (detach)

Quando efetuamos várias alterações, não seguidas, ao estado de um objeto, para evitar a execução de vários updates, podemos passar a entidade para o estado detached (não gerido). No fim das alterações devemos passar a entidade para o estado managed usando o método merge.

```
public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPAPessoaPU");
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();
    Pessoa p1 = em.find(Pessoa.class, 1);
    em.detach(p1);
    p1.alterarMorada("Gondomar");
    em.getTransaction().commit();
    System.out.println("ID gerado: " + p1.getId());
    em.close();
    emf.close();
}
```

A alteração de morada não é persistida na base de dados

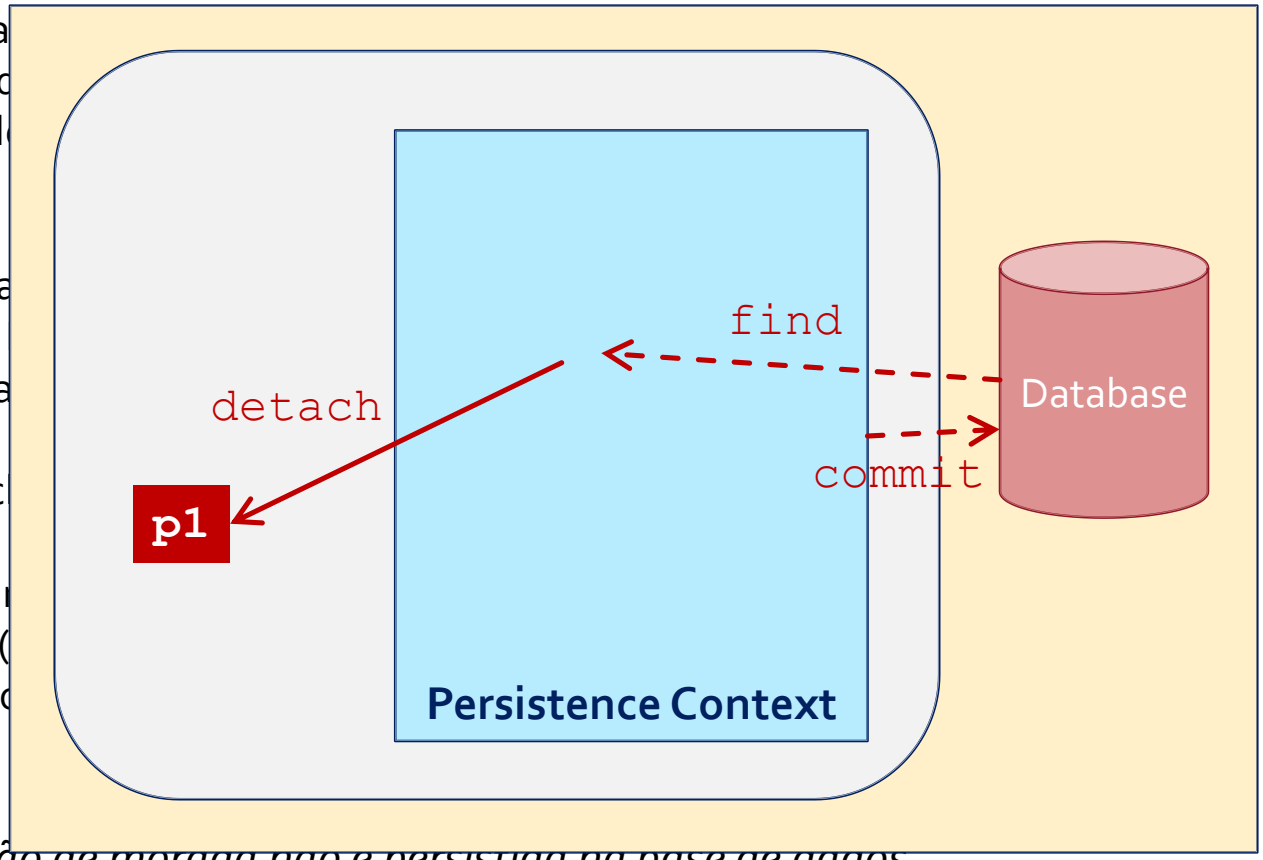
Saída produzida pelo programa: ID gerado: 1

Entity Lifecycle

4. Estado Detached (detach)

Quando efetuamos várias altera
execução de vários updates, poc
gerido). No fim das alterações d
método merge.

```
public static void main(String[] a
    EntityManagerFactory emf =
    EntityManager em = emf.crea
    em.getTransaction().begin();
    Pessoa p1 = em.find(Pessoa.c
    em.detach(p1);
    p1.alterarMorada("Gondomar
    em.getTransaction().commit(
    System.out.println("ID gerad
    em.close();
    emf.close();
}
```



A alteração de morada não é persistida na base de dados

Saída produzida pelo programa: ID gerado: 1

Entity Lifecycle

5. Estado Managed (merge) – funcionamento incorreto

O método merge leva uma entidade como parâmetro, cria uma nova entidade com os valores da base de dados, coloca essa entidade no estado managed, atribui-lhe os atributos diferentes da entidade parâmetro e retorna a entidade criada.

```
public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPAPessoaPU");
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();
    Pessoa p1 = em.find(Pessoa.class, 1);
    em.detach(p1);
    em.merge(p1);
    p1.alterarMorada("Gondomar");
    em.getTransaction().commit();
    System.out.println("ID gerado: " + p1.getId());
    em.close();
    emf.close();
}
```

O objeto criado pelo método merge é retornado pelo mesmo método merge, mas neste programa não é aproveitado.
Após o merge a entidade p1 continua no estado detached e por isso alterações no seu estado ***não são persistidas*** *na base de dados.*

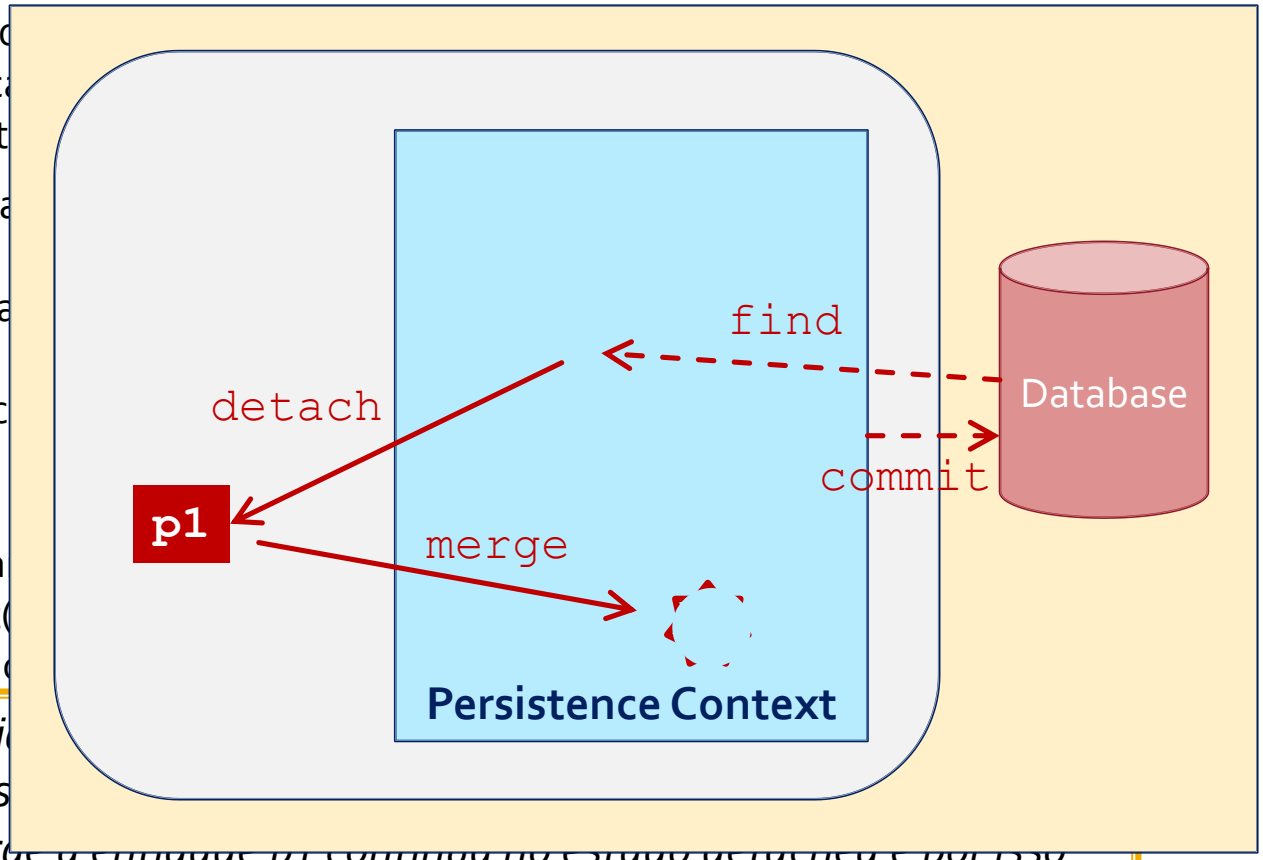
Após o merge a
merge é retornado pelo mesmo método merge, mas neste programa não é aproveitado.

Entity Lifecycle

5. Estado Managed (merge) – funcionamento incorreto

O método merge leva uma entid
valores da base de dados, coloc
diferentes da entidade parâmet

```
public static void main(String[] a
    EntityManagerFactory emf =
    EntityManager em = emf.crea
    em.getTransaction().begin();
    Pessoa p1 = em.find(Pessoa.c
    em.detach(p1);
    em.merge(p1);
    p1.alterarMorada("Gondoma
    em.getTransaction().commit(
    System.out.println("ID gerad
    em.close();
    emf.close();
}
```



O objeto cri
merge, mas
Após o merge a entidade p1 continua no estado detached e por isso
alterações no seu estado ***não são persistidas*** *na base de dados.*

Após o merge a
merge é retornado pelo mesmo método merge, mas neste programa não é aproveitado.

Entity Lifecycle

6. Estado Managed (merge) – funcionamento correto

A entidade criada pelo método merge é colocada no estado managed, em seguida o seu estado é alterado sendo marcada como dirty, e as modificações são atualizadas na base de dados no commit da transação.

```
public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPAPessoaPU");
    EntityManager em = emf.createEntityManager();
    em.getTransaction().begin();
    Pessoa p1 = em.find(Pessoa.class, 1);
    em.detach(p1);
    p1 = em.merge(p1);
    p1.alterarMorada("Gondomar");
    em.getTransaction().commit();
    System.out.println("ID gerado: " + p1.getId());
    em.close();
    emf.close();
}
```

Após o merge p1 passa a referenciar a instância criada e colocada no estado ma
*Assim alterações no seu estado **são persistidas** na base de dados.*

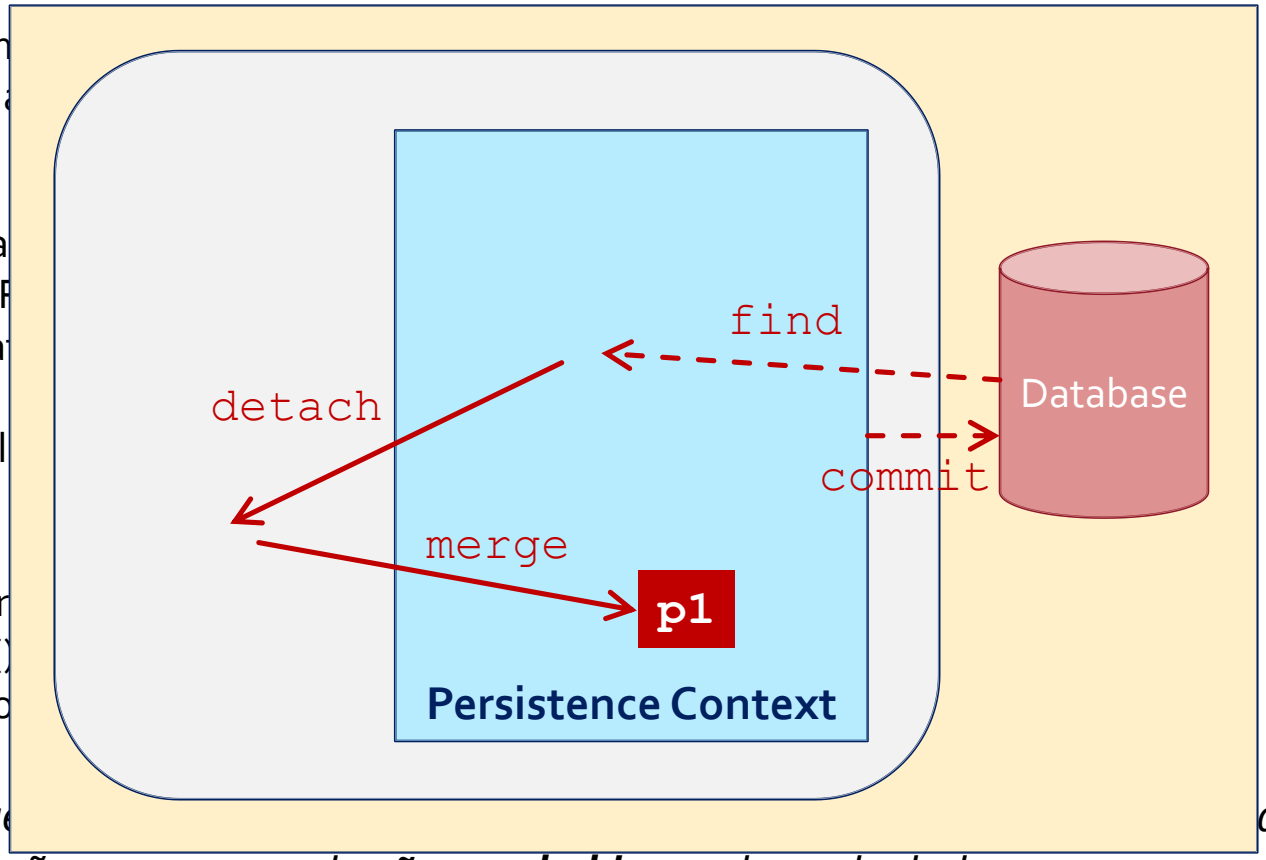
Saída produzida pelo programa: ID gerado: 1

Entity Lifecycle

6. Estado Managed (merge) – funcionamento correto

A entidade criada pelo método n
estado é alterado sendo marcad
dados no commit da transação.

```
public static void main(String[] a
    EntityManagerFactory emf = F
    EntityManager em = emf.creat
    em.getTransaction().begin();
    Pessoa p1 = em.find(Pessoa.cl
    em.detach(p1);
    p1 = em.merge(p1);
    p1.alterarMorada("Gondomar
    em.getTransaction().commit()
    System.out.println("ID gerado
    em.close();
    emf.close();
}
```



Após o merge

*Assim alterações no seu estado **são persistidas** na base de dados.*

Saída produzida pelo programa: ID gerado: 1

Entity Lifecycle

- **Ciclo de vida** das entidades
  - Estados: New/Transient, Managed, Detached, Removed
  - Eventos, transições de estado
- **Entity Manager**: clear, close, contains, detach, find, flush, merge, persist, refresh, remove
- **EntityTransaction**: begin, commit, rollback, isActive
- **Persistence Context**