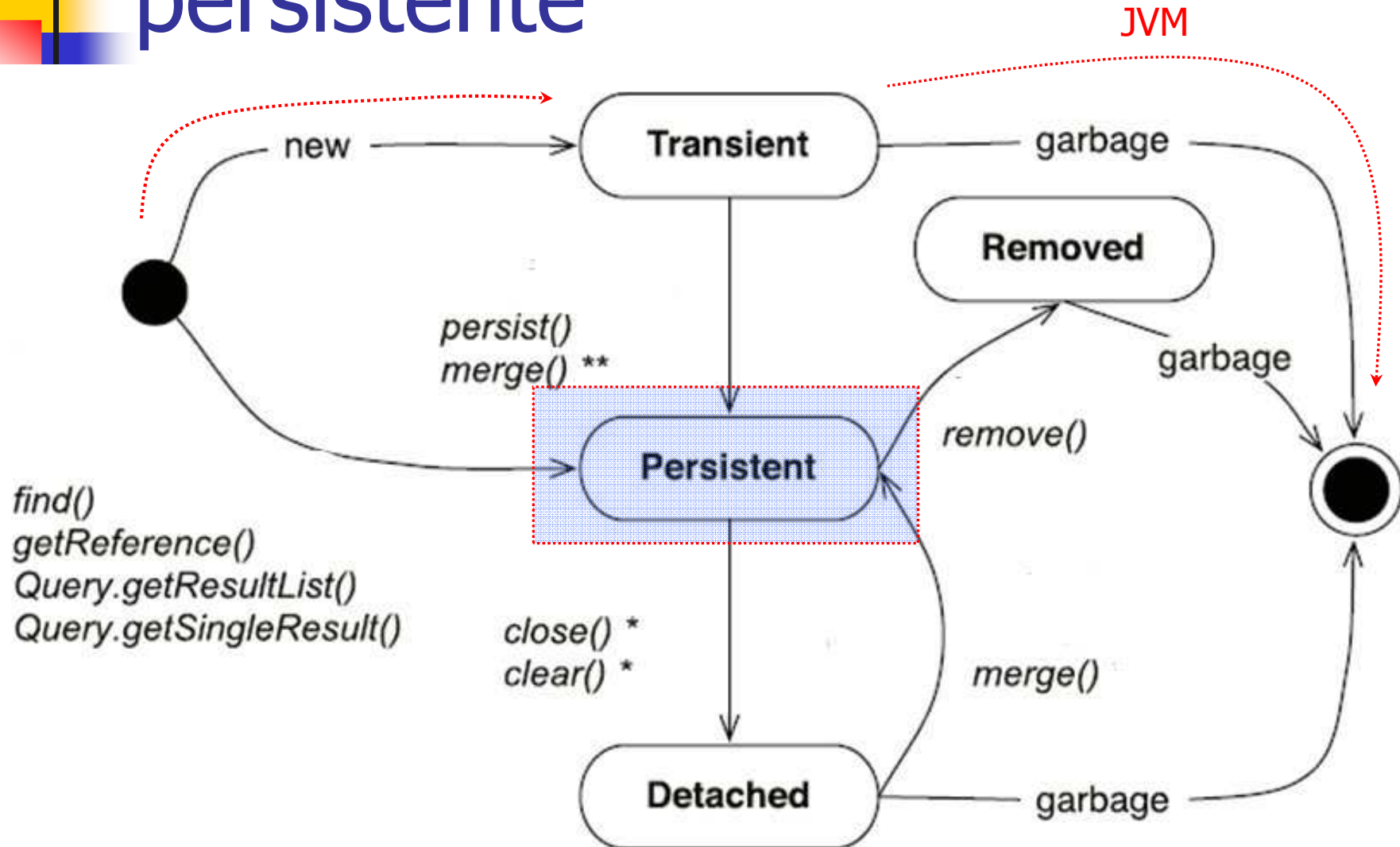




Gestión de objetos persistentes en JPA

Sistema de Persistencia de
Objetos

Ciclo de vida un objeto persistente





Control del ciclo de vida

- Se gestiona desde un EntityManager
 - Es el gestor de persistencia de JPA
- El EntityManager (la sesión) es el ámbito de persistencia
 - El ciclo de vida tiene lugar en la memoria de la JVM
 - Un objeto “está en sesión” cuando está en **Persistent**
- La sesión es una caché de primer nivel que:
 - Garantiza la identidad java y la identidad DB
 - No habrá varios objetos en sesión representando la misma fila
 - JPA (hibernate) optimiza el SQL para minimizar tráfico a la BBDD
 - Dirty-checking
 - Write-behind



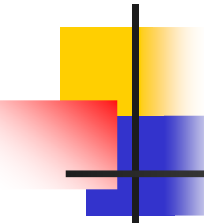
Dentro del contexto de persistencia(EntityManager) ...

- Se lleva a cabo una unidad de trabajo (UOK)
- Al final de la unidad de trabajo se sincroniza con la BBDD
- La sesión lleva traza de todos los cambios hechos a los objetos en memoria durante la unidad de trabajo
- Al hacer COMMIT o FLUSH hibernate organiza las actualizaciones para optimizar el rendimiento
- La identidad se garantiza porque una fila de la BBDD solo se carga una vez y es representada por un único objeto java por contexto de persistencia
 - Pero puede haber muchos contextos simultáneos...



Estados de persistencia

- Transient
 - Un objeto recién creado que no ha sido enlazado con el gestor de persistencia (solo existe en memoria de la JVM)
- Persistent
 - Un objeto enlazado con la sesión
 - Todos los cambios que se le hagan serán persistentes
- Detached
 - Un objeto persistente que sigue en memoria después de que termina la sesión: existe en java y en la BDD
- Removed
 - Un objeto marcado para ser eliminado de la BBDD: existe en java y se borrará de la BDD al terminar la sesión



```
Category newCategory = new Category();
newCategory.setName("Perifericos"); / <--- Transient
```

```
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
    em.persist(newCategory); / <-- Persistent
tx.commit();
em.close();
```

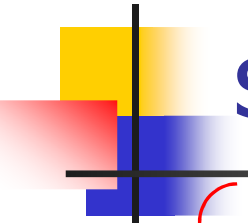
```
newCategory.setName("Perifericos para PC"); /<-- Detached
```

```
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
    em.merge(newCategory); / <-- Persistent
tx.commit();
em.close();
```

```
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
    em.remove(newCategory); / <-- Removed
tx.commit();
em.close();
```

```
newCategory.setName("Perifericos para Apple"); / <-- Transient
```

Ámbito de identidad garantizada sólo dentro del contexto



```
Session session1 = sessionFactory.openSession();
Transaction tx1 = session1.beginTransaction();

// Load Item with identifier value "1234"
Object a = session1.get(Item.class, new Long(1234) );
Object b = session1.get(Item.class, new Long(1234) );

( a==b ) // True, persistent a and b are identical

tx1.commit();
session1.close();
```

// References a and b are now to an object in detached state

```
Session session2 = sessionFactory.openSession();
Transaction tx2 = session2.beginTransaction();

Object c = session2.get(Item.class, new Long(1234) );

( a==c ) // False, detached a and persistent c are not identical

tx2.commit();
session2.close();
```



Identidad fuera de la sesión

...

```
session2.close();
```

```
Set allObjects = new HashSet();
```

```
allObjects.add(a);
```

```
allObjects.add(b);
```

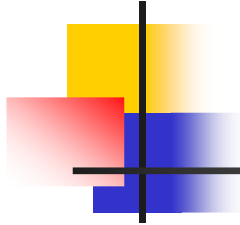
```
allObjects.add(c);
```

Siguiendo con el ejemplo anterior:

¿ Cuánto vale `allObjects.size()` ?

`Set()` depende de `equals()` → hay que implementar `equals()` en todos los objetos que se vayan a guardar en colecciones para estar seguros de lo que hace

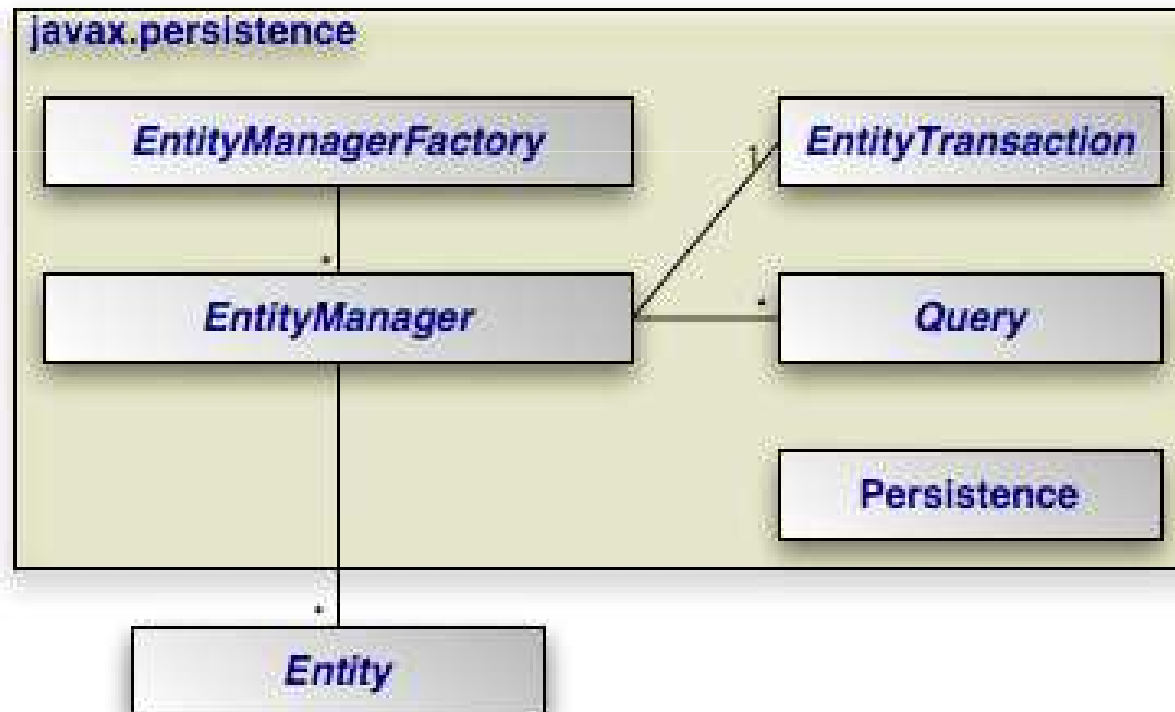
Sincronización de la sesión y la BBDD



- Ocurre lo más tarde posible:
 - Cuando se hace COMMIT a una transacción
 - Antes de que se ejecute una consulta
 - Cuando se llama `entityManager.flush()`
- Se puede modificar el comportamiento
 - `entityManager.setFlushMode(...)`
 - `FlushMode.AUTO`
 - `FlushMode.COMMIT`
 - `FlushMode.MANUAL` ← sólo hibernate

API de EntityManager

- Factoría de Consultas y Transacciones



APIs JPA

EntityManager

- clear()
- close()
- contains(Object)
- createNamedQuery(String)
- createNativeQuery(String)
- createNativeQuery(String, Class)
- createNativeQuery(String, String)
- createQuery(String)
- find(Class<T>, Object) <T>
- flush()
- getDelegate()
- getFlushMode()
- getReference(Class<T>, Object) <T>
- getTransaction()
- isOpen()
- joinTransaction()
- lock(Object, LockModeType)
- merge(T) <T>
- persist(Object)
- refresh(Object)
- remove(Object)
- setFlushMode(FlushModeType)

EntityManagerFactory

- close()
- createEntityManager()
- createEntityManager(Map)
- isOpen()

EntityTransaction

- begin()
- commit()
- getRollbackOnly()
- isActive()
- rollback()
- setRollbackOnly()

Query

- executeUpdate()
- getResultList()
- getSingleResult()
- setFirstResult(int)
- setFlushMode(FlushModeType)
- setHint(String, Object)
- setMaxResults(int)
- setParameter(int, Object)
- setParameter(int, Calendar, TemporalType)
- setParameter(int, Date, TemporalType)
- setParameter(String, Object)
- setParameter(String, Calendar, TemporalType)
- setParameter(String, Date, TemporalType)

Ámbito de persistencia

```
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();

tx.begin();
Item item = em.find(Item.class, new Long(1234));
tx.commit();

Item.setDescription(...);

tx.begin();
User user = em.find(User.class, new Long(3456));
user.setPassword("secret");
tx.commit();

em.close();
```

Item todavía persistente,
se salva **aquí**



Gestionando objetos...

- Inicio de una unidad de trabajo
- Fin de la unidad de trabajo
- Hacer un objeto persistente
- Cargar un objeto persistido
- Modificar un objeto persistente
- Hacer transient objeto persistente
- Hacer detached todos los objetos del contexto
- Hacer persistente un objeto detached
- Hacer transient un detached
- Merge con un objeto detached
- Merge con un objeto detached: algoritmo JPA
- Flush del contexto de persistencia

Gestionando objetos

■ Inicio de la unidad de trabajo

```
EntityManagerFactory emf =
    Persistence.createEntityManagerFactory("caveatemptorDatabase");
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
```

■ Fin de la unidad de trabajo

```
tx.commit();
em.close();
```

```
tx.rollback();
em.close();
```

nov-08

Gestión más correcta de la
sesión y transacción:
control de las excepciones

alb@uniovi.es

```
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
try {
    tx.begin();
    //do some work...
    . . .
    tx.commit();
}
catch(PersistenceException pe){
    tx.rollback();
    throw pe;
}
finally{
    em.close();
}
```

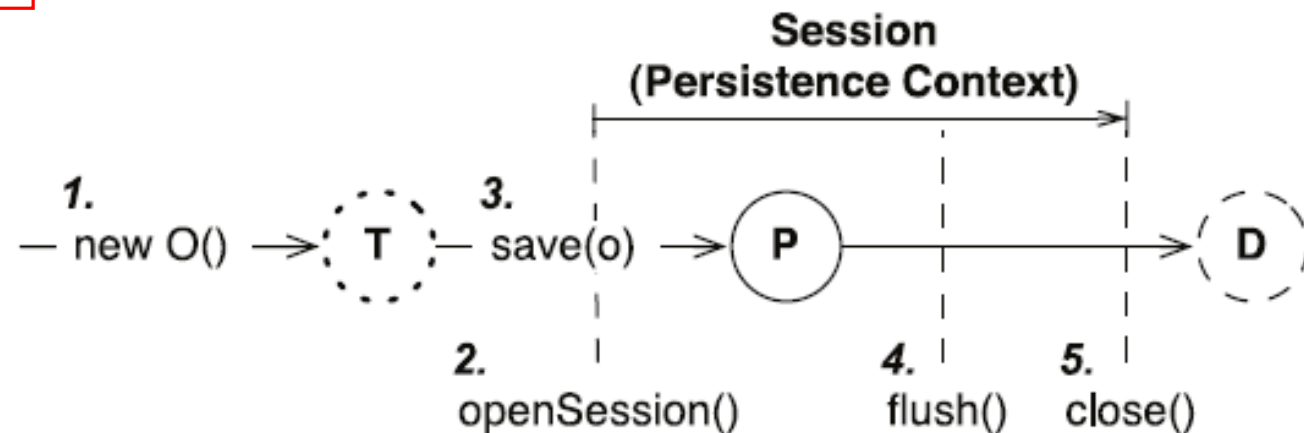
Hacer objeto persistente

```
Item item = new Item();  
item.setName("Playstation3 incl. all accessories");  
item.setEndDate( ... );
```

```
EntityManager em = emf.createEntityManager();  
EntityTransaction tx = em.getTransaction();  
tx.begin();
```

```
em.persist(item);
```

```
tx.commit();  
em.close();
```



Cargar objeto persistente

```
EntityManager em = emf.createEntityManager();  
EntityTransaction tx = em.getTransaction();  
tx.begin();
```

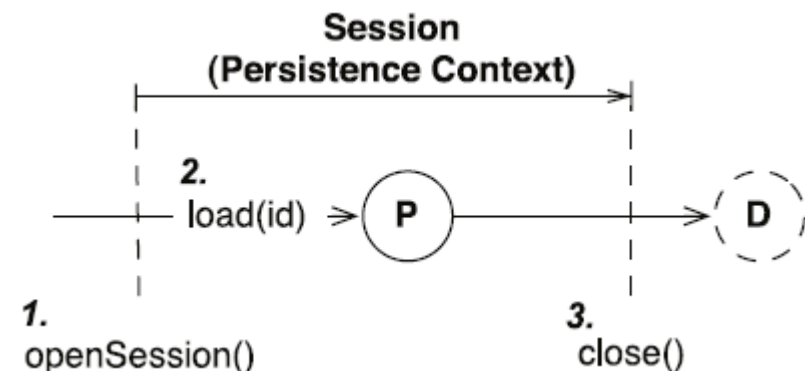
```
Item item = em.find(Item.class, new Long(1234));  
Item item = em.getReference(Item.class, new Long(1234));
```

```
tx.commit();  
em.close();
```

¿ **find()** o **getReference()** ?

Si no existe en BDD:

- **find()** devuelve **null**
- **getReference()** devuelve un proxy
 - Que puede lanzar **EntityNotFoundException**

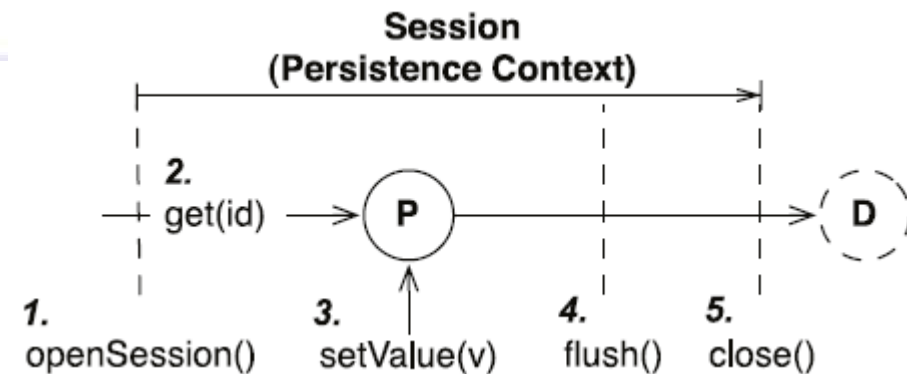


Modificar objeto persistente

```
EntityManager em = emf.createEntityManager();  
EntityTransaction tx = em.getTransaction();  
tx.begin();
```

```
Item item = em.find(Item.class, new Long(1234));  
item.setDescription(...);
```

```
tx.commit();  
em.close();
```

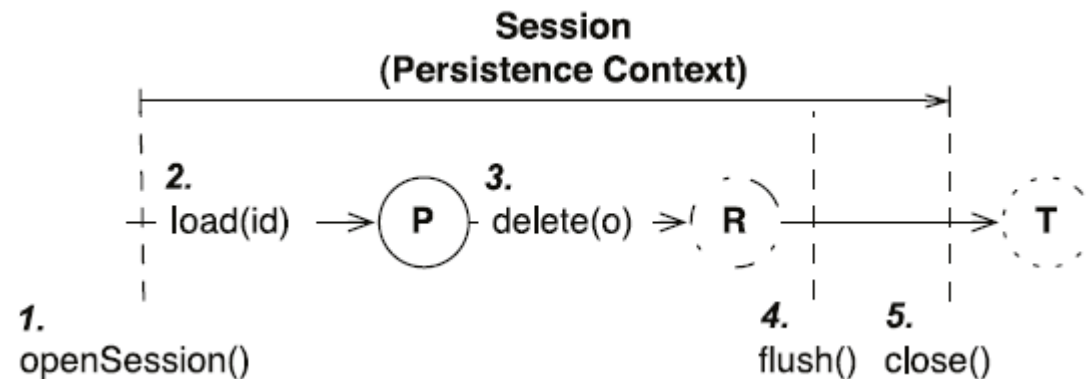


Hacer transient objeto persistente

```
EntityManager em = emf.createEntityManager();  
EntityTransaction tx = em.getTransaction();  
tx.begin();
```

```
Item item = em.find(Item.class, new Long(1234));  
em.remove(item);
```

```
tx.commit();  
em.close();
```





Hacer detached todos los objetos del contexto

```
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();

Item item = em.find(Item.class, new Long(1234));
em.clear();
item.setDescription(...); // Detached entity instance!

tx.commit();
em.close();
```

Item **no** se sincroniza al commit()

Hacer persistente un objeto detached

```
item.setDescription(...); // Loaded in previous Session
```

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

```
EntityTransaction tx = em.getTransaction();
```

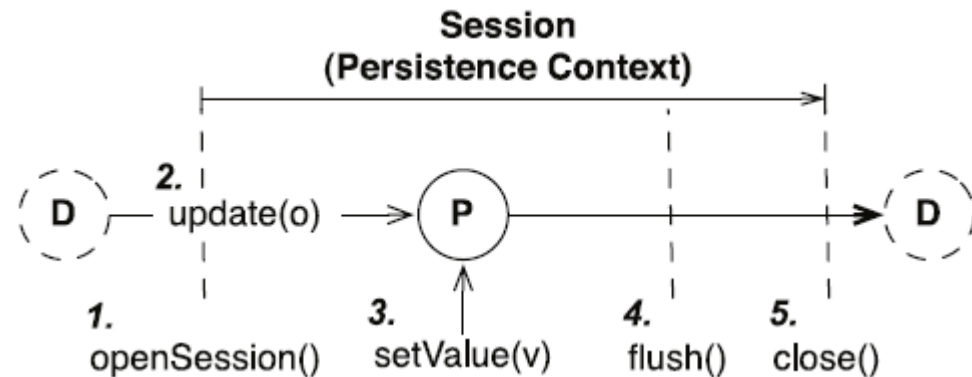
```
tx.begin();
```

```
em.merge(item);
```

```
item.setEndDate(...);
```

```
tx.commit();
```

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





Hacer transient un detached

```
EntityManager em = emf.createEntityManager();  
EntityTransaction tx = em.getTransaction();  
tx.begin();  
  
em.remove(item);  
  
tx.commit();  
em.close();
```

Detached → no está en sesión pero sí en BDD (y en JVM)

Transient → no está en sesión ni en BDD (pero sí en JVM)

Merge con un objeto detached

```
item.setDescription(...); // Detached entity instance!
```

```
EntityManager em2 = emf.createEntityManager();  
EntityTransaction tx2 = em2.getTransaction();  
tx2.begin();
```

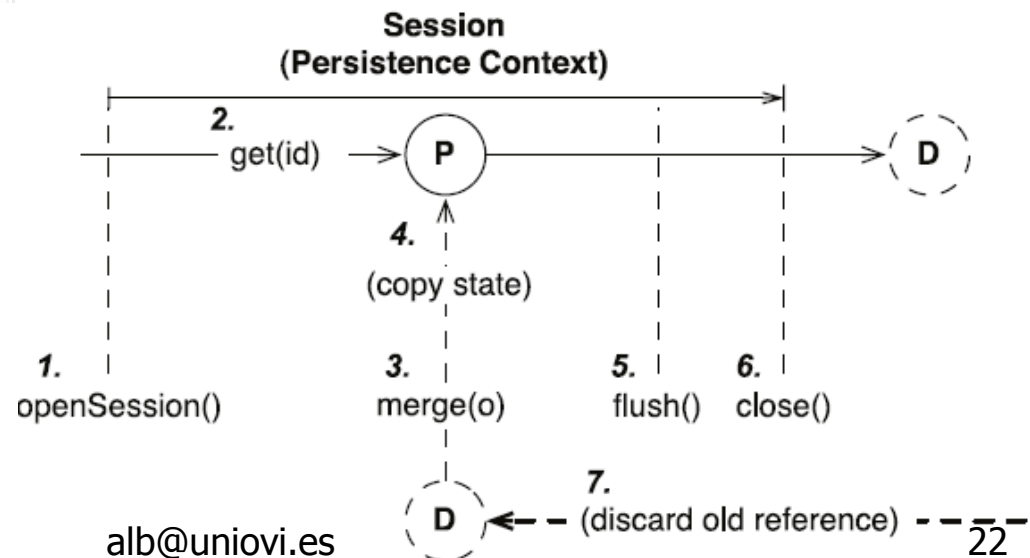
```
Item mergedItem = (Item) em2.merge(item);
```

```
{ item == mergedItem } // false
```

```
tx2.commit();  
em2.close();
```

item **no está** en contexto
mergedItem sí está

Devuelve un
objeto
nuevo





Merge con un objeto detached: algoritmo JPA

- Si existe otro objeto persistente con misma identidad BDD
 - Copiar detached en persistente
- Si existe en BDD
 - cargar y actualizar datos con los del detached
- Si no esta en BDD
 - Es objeto nuevo, se hace persistente

Flushing el contexto de persistencia

- Cuando EntityTransaction committed
- “Antes” de ejecutar una query
- Llamando a **em.flush()**

```
EntityManager em = emf.createEntityManager();
em.setFlushMode(FlushModeType.COMMIT);
EntityTransaction tx = em.getTransaction();
tx.begin();

Item item = em.find(Item.class, new Long(1234));
item.setDescription(...);
List result = em.createQuery(...).getResultList();

tx.commit();
em.close();
```

FlushMode.AUTO

- Commit
- Query
- em.flush()**

FlushMode.COMMIT

- Commit
- em.flush()**