

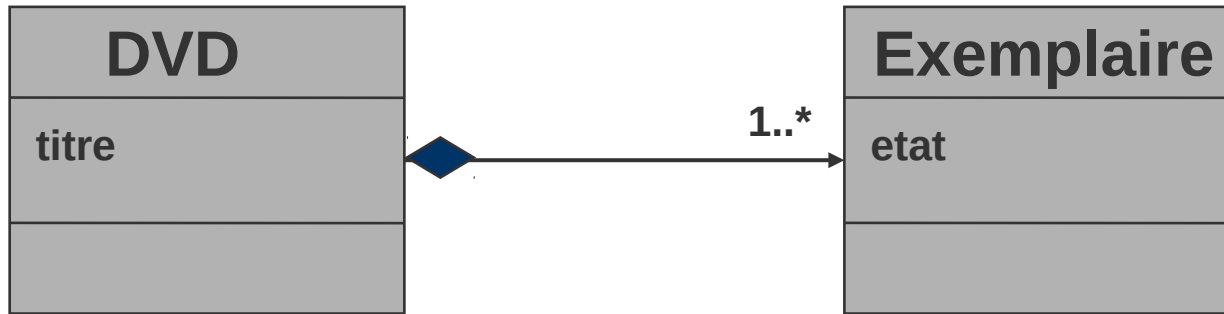
Cascading et composition

Plan

- Contraintes de cycle de vie
- Persistance transitive
- Composition

Contraintes de cycle de vie

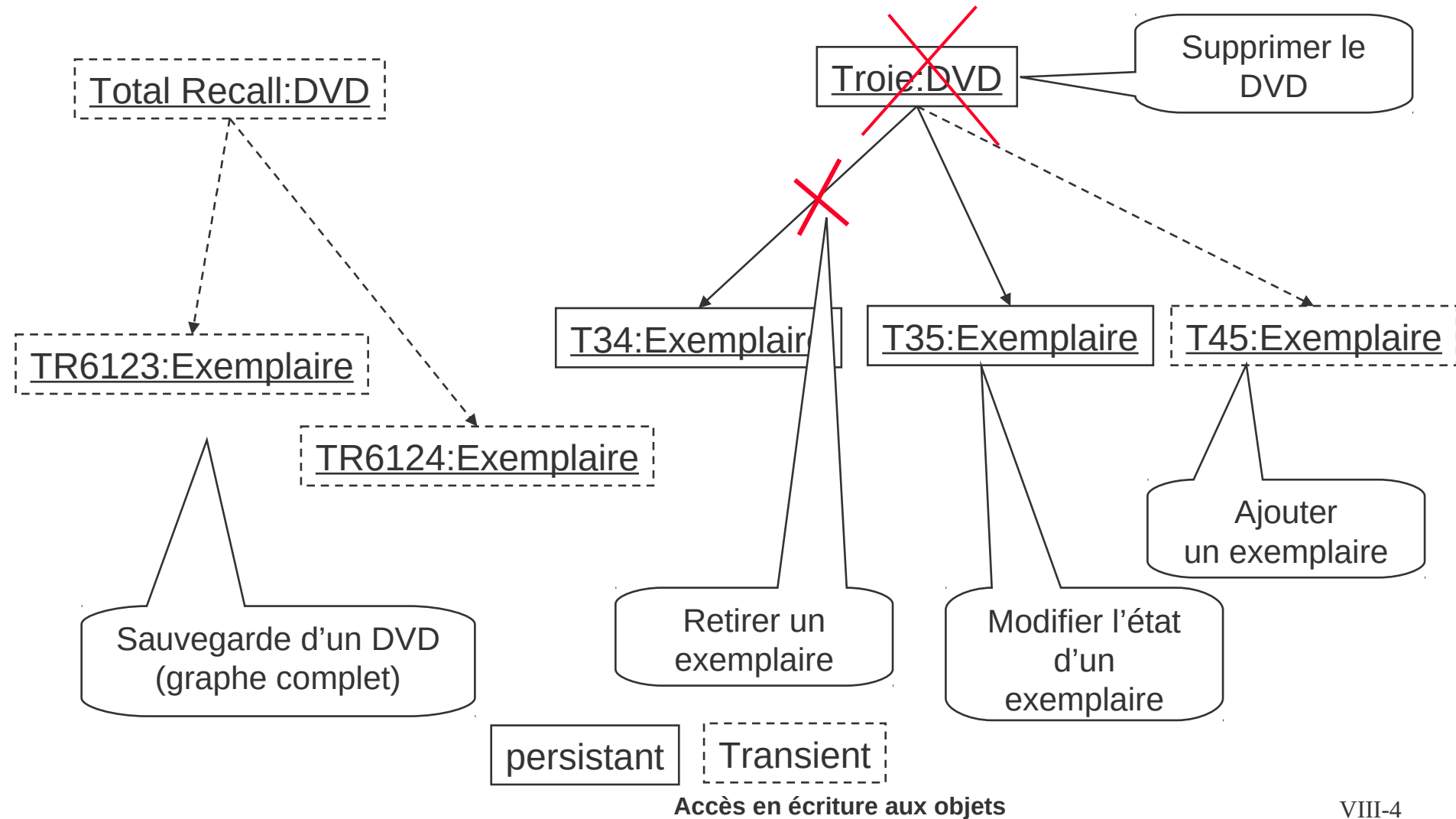
Généralités



- Composition
 - Un composant peut exister par lui-même.
 - La suppression du DVD entraîne celle des exemplaires.
- Composition stricte
 - Un composant n'a d'existence que dans son composé.

Contraintes de cycle de vie

différents cas




Persistence Transitive

la sauvegarde

- Sauvegardons un nouveau DVD

```
EntityManager em = DBHelper.getFactory().createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin()
DVD dvd = new DVD();
dvd.setTitre("Total Recall");
dvd.setDuree("2h05");
dvd.setTexte("Sur Mars, la révolte...");
Exemplaire e1 = new Exemplaire();
e1.setCodeBarre("TR-7845-5684");
dvd.addExemplaire(e1);
em.persist(dvd);
tx.commit();
em.close();
```



SEVERE: Could not synchronize database state with session
org.hibernate.**TransientObjectException**: object references an unsaved transient instance –
save the transient instance before flushing: com.tsystems.etechno.j12.exemple.metier.Exemplaire
at org.hibernate.engine.Foreign

Persistence Transitive

la suppression

- Supprimons un Dvd existant

```
EntityManager em = DBHelper.getFactory().createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin()
DvD dvd = (DvD)s.get(DvD.class,new Long(1));
em.remove(dvd);
tx.commit();
em.close();
```

Les exemplaires
ne sont pas
supprimés !

Mediatheque / texemplaire: 3 Records (3 retrieved)

	ID	code	etat	IDItem
▶	1	pl-1452-gt		1
	2	pl-1427-gt		1
	3	pl-7584-gt		1

Persistence Transitive

JPA propriété cascade

L'attribut ***cascade*** permet de d'indiquer à l'*EntityManager* comment il propage une opération de persistance aux instances liées

Lorsqu'une opération *persist*, *merge*, *remove*, ou *refresh* est appliquée à un bean entité, la même opération peut être propagée sur les beans entités liées par l'association :

@OneToMany(cascade={CascadeType.ALL})

- la valeur *ALL* signifie que toute opération *persist*, *merge*, *remove*, *refresh* est propagée aux beans entités liés – autres valeurs possibles: *PERSIST*, *MERGE*, *REMOVE*, *REFRESH*, *DETACH*
- par défaut, aucune opération n'est propagée
- L'attribut *orphanRemoval=true* déclenche la suppression de l'entité associée lorsque sa référence est supprimée (Par exemple, la méthode *remove()* d'une Collection)

@OneToMany(cascade={CascadeType.ALL},orphanRemoval=true)

Composition stricte

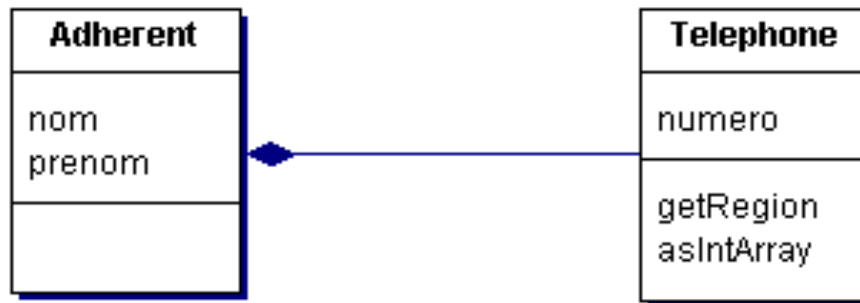
Objet entité et objet valeur

- On distingue deux catégories d'objets en Hibernate :
 - Les objets-entités
 - Possède une identité (clé primaire)
 - Ont un cycle de vie propre
 - Les objets-valeurs
 - N'ont pas d'identité
 - Suivent le cycle de vie de l'objet englobant
 - Une ligne d'une table correspond à plusieurs objets.

Composition stricte

Exemple

- *Adherent* est un objet-entité.
- *Telephone* est un objet-valeur qui dépend d'adhérent.
- C'est une relation de composition stricte.



Composition stricte la table

TAdherent

Colonne
représentant la
classe Téléphone

	ID	nom	prenom	tel	email	adresse	mdp
<input type="checkbox"/>	1	Marytin	Jean	01.23.25.45.26	jm@provider.com		jmart
<input type="checkbox"/>	2	Valron	Annette	01.47.25.65.12	annette54@provider.com	test adresse	aval
<input type="checkbox"/>	3	Frament	Jean	01.78.36.54.45	jframent@net.fr		jfram
<input type="checkbox"/>	4	Emtec	Henri	02.54.23.65.12	hemtec@net.fr		htec
<input type="checkbox"/>	*						

Composition stricte

JPA

- JPA autorise la persistance d'attributs, instances de classes non persistantes
 - les propriétés de ces instances sont alors mappées sur la même table que celle de la classe entité
 - la classe non persistante doit être annotée avec ***@Embeddable***
 - le champ ou la propriété instance de la classe non persistante doit être annotée ***@Embedded***

Composition stricte JPA

```
@Embeddable public class Adresse implements Serializable
```

```
@Column(name="VOIE")
```

```
private String voie;
```

```
@Column(name="CODE")
```

```
private String codePostal;
```

```
@Column(name="VILLE")
```

```
private String ville;
```

```
public Adresse () {}
```

```
    public String getVoie() { return voie;}
```

```
    public void setVoie(String voie) { this.voie=voie; }
```

```
    public long getCodePostal() { return codePostal; }
```

```
    public void setCodePostal(String code) { codePostal=code; }
```

```
    public long getVille() { return ville; }
```

```
    public void setVille(String ville) {ville=ville; }
```

```
}
```

Composition stricte JPA

```
@Entity public class Client implements Serializable {
    private ClientPK pk;
    private String nom;
    private Adresse adresse;

    @Id @GeneratedValue
    public ClientPK getPk() { return pk; }
    public void setPk(ClientPK pk) { this.pk = pk; }

    public String getNom() { return pk.getNom(); }
    public void setNom(String nom) { pk.setNom(nom); }

    @Embedded
    @AttributeOverride(name="codePostal",
        column=@Column(name="CODE_POSTAL"))
    public Adresse getAdresse() { return adresse; }
    public void setAdresse(Adresse ad) { adresse=ad; }
}
```

Exercice

- Exercice 7 : La persistance transitive