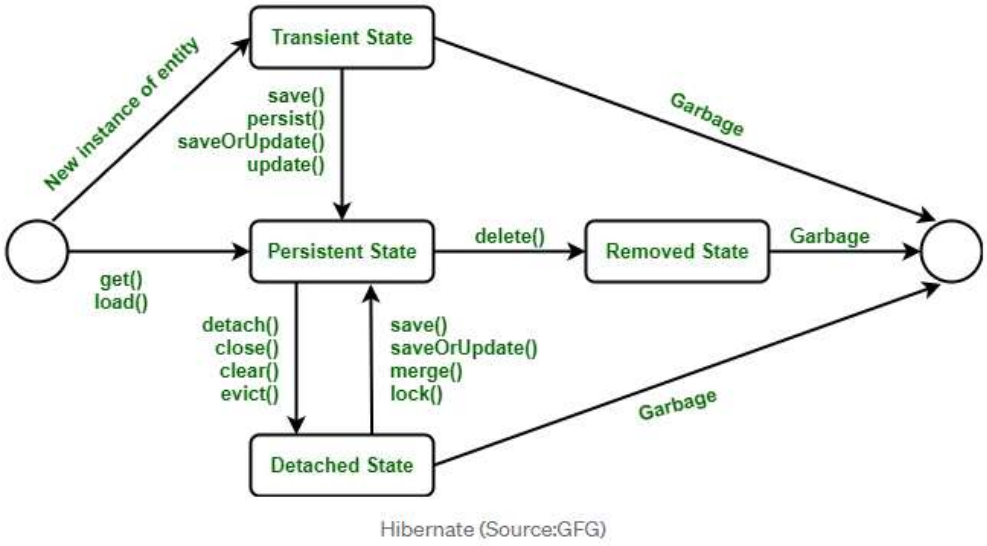
# Hibernate Entity Lifecycle

The lifecycle of hibernate session can be divided into three states which are:

1. Transient
2. Persistent or Managed
3. Detached
4. Removed



Hibernate (Source:GFG)

### 1.1. *Transient*

Transient entities exist in heap memory as normal Java objects. Hibernate does not manage transient entities. The persistent context does not track the changes done on them.

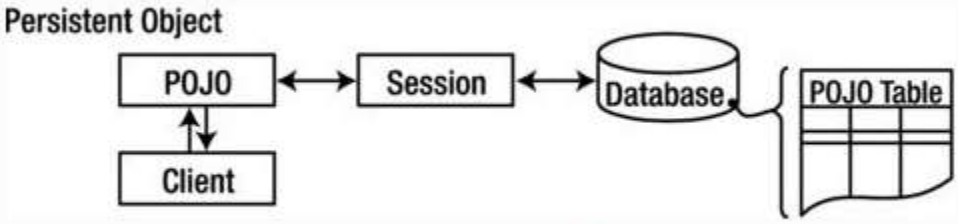
- In simple words, a transient entity has neither any representation in the datastore nor in the current *Session*.
- A transient entity is simply a POJO without any identifier.

```
Transient EmployeeEntityEmployeeEntity employee = new EmployeeEntity();
```

***Persistent* or *Managed***

Persistent entities exist in the database, and Hibernate's **persistent context tracks all the changes done on the persistent entities** by the client code.

A persistent entity is mapped to a specific database row, identified by the *ID* field. Hibernate's current running *Session* is responsible for tracking all changes done to a managed entity and propagating these changes to database.




Persistent objects are maintained by Hibernate

We can get persistent entity in either of two ways:

- Load the entity using *get()* or *load()* method.
- Persist the transient or detached entity using *persist()*, *save()*, *update()* or *saveOrUpdate()* methods.

```
EmployeeEntity employee = session.load(1);

or

EmployeeEntity employee = new EmployeeEntity();
session.save(employee);
```

### 1.3. Detached

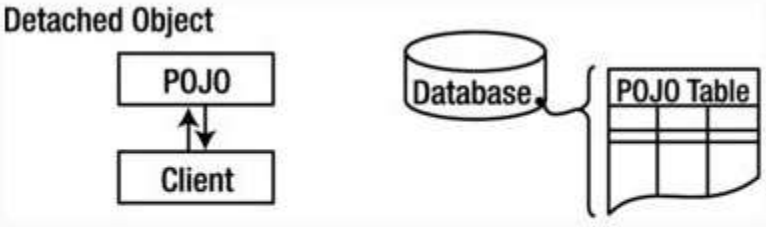
**Detached entities have a representation in the database but these are currently not managed by the *Session***. Any changes to a detached entity will not be reflected in the database, and vice-versa.

A detached entity can be created by closing the session that it was associated with, or by evicting it from the session with a call to the session's evict() method.

Detaching an entity from Sessionsession.close();

//or

session.evict(entity);




Detached objects exist in the database but are not maintained by Hibernate

Note that in order to make a persistent entity from a detached entity, the application must re-attach it to a valid Hibernate *Session*. A detached instance can be associated with a new Hibernate session when your application calls one of the load(), refresh(), merge(), update(), or save() methods on the new session with a reference to the detached object.

### 1.4. Removed

Removed entities are objects that were being managed by Hibernate (persistent entities, in other words) and now those have been passed to the session's remove() method.

When the application marks the changes held in the *Session* as to be committed, the entries in the database that correspond to removed entities are deleted.

```
session.remove(employee);
```

**Conclusion**

1. The newly created POJO object will be in the transient state. The transient entity doesn't represent any row of the database i.e. not associated with any session object. It's a plain simple java object.
2. A persistent entity represents one row of the database and is always associated with some unique hibernate session. Changes to persistent objects are tracked by hibernate and are saved into the database when commit calls happen.
3. Detached entities are those who were once persistent in the past, and now they are no longer persistent. To persist changes done in detached objects, you must re-attach them to hibernate session.
4. Removed entities are persistent objects that have been passed to the session's remove() method and soon will be deleted as soon as changes held in the session will be committed to the database.