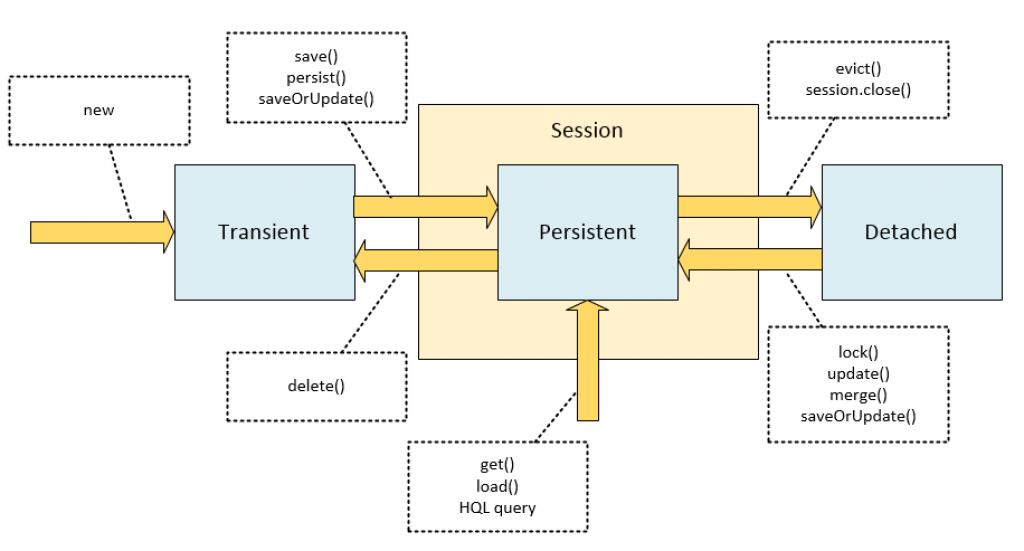
Update -> Hibernate method not provided by JPA.

Issue happens if we “A different object with the same identifier value was already associated with the session”. Here I added Account account1 = session.get(Account.class, 3);

1. updateWithTransient() -> update will throw an exception.
2. Issue of updating detached object can be solved using merge.

|  |  |  |
| --- | --- | --- |
|  | Update (Hibernate) | Merge (JPA) |
| Transient | It will fire update query. Row not found exception | Creates persistent entity. (Select query and then insert query—assigned integer. Similar for dettached)  Correct , since object is not in persistent context direct insert 🡪 auto generated identifier |
| **Dettached** (A persistent object with the same identifier value doesnot exist) | It will make detached object to persistent state. | It will hit database get persistent object and then update persistent object with updated values of detached object. |
| Dettached (A persistent object with the same identifier value exist) | Throws an Exception | It will reuse the persistent object and then update persistent object with updated values of detached object. And then update persistent object to database via update SQL. |



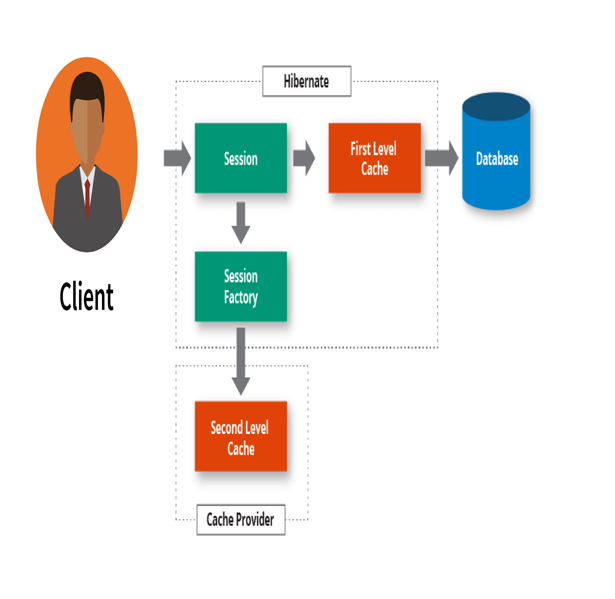
Load v/s get

| **get()** | **load()** |
| --- | --- |
| This method gets the data from the database as soon as it is called. | This method returns a proxy object and loads the data only when it is required. |
| The database is hit every time the method is called. | The database is hit only when it is really needed and this is called Lazy Loading which makes the method better. |
| The method returns null if the object is not found. | The method throws ObjectNotFoundException if the object is not found. |

//commit has to go in hand with session.save, persist, update e.t.c

| **First Level Cache** | **Second Level Cache** |
| --- | --- |
| This is local to the Session object and cannot be shared between multiple sessions. | This cache is maintained at the SessionFactory level and shared among all sessions in Hibernate. |
| This cache is enabled by default and there is no way to disable it. | This is disabled by default, but we can enable it through configuration. |
| The first level cache is available only until the session is open, once the session is closed, the first level cache is destroyed. | The second-level cache is available through the application’s life cycle, it is only destroyed and recreated when an application is restarted. |

If an entity or object is loaded by calling the get() method then Hibernate first checked the first level cache, if it doesn’t find the object then it goes to the second level cache if configured. If the object is not found then it finally goes to the database and returns the object, if there is no corresponding row in the table then it returns nul



| **save()** | **saveOrUpdate()** |
| --- | --- |
| save() generates a new identifier and INSERT record into a database | Session.saveOrUpdate() can either INSERT or UPDATE based upon existence of a record. |
| The insertion fails if the primary key already exists in the table. | In case the primary key already exists, then the record is updated. |
| The return type is Serializable which is the newly generated identifier id value as a Serializable object. | The return type of the saveOrUpdate() method is void. |
| This method is used to bring only a transient object to a persistent state. | This method can bring both transient (new) and detached (existing) objects into a persistent state. It is often used to re-attach a detached object into a Session |