Hibernate-3

Primary key auto generator-

- Hibernate will generate primary key auto generated internally we don’t need to pass id manually.
- Why?
- Suppose in booking movie ticket, there are some field such as id, name, time, etc. if first user pass the id as 1, name as ram, time as 2 pm and second user pass the id as 1, name as shyam, time as 2 pm, second user get the exception as primary key must be unique, He could not book the ticket.

- To overcome this problem, we should go for primary key auto generator.

options to generate primary keys

- The JPA specification supports 4 different primary key generation strategies which generate the primary key values programmatically or use database features, like auto-incremented columns or sequences. The only thing you have to do is to add the *@GeneratedValue* annotation to your primary key attribute and choose a generation strategy.
- 1.1 GenerationType.AUTO
- 1.2 GenerationType.IDENTITY
- 1.3 GenerationType.SEQUENCE
- 1.4 GenerationType.TABLE

GenerationType.AUTO

- GenerationType.AUTO is the default strategy.
- This lets the Hibernate to choose the best strategy based on the database dialect. For most of the common databases, it picks GenerationType.SEQUENCE.

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Integer id;
```

1.2 GenerationType.IDENTITY

- GenerationType.IDENTITY lets the database to generate this value, mostly by an auto-increment logic

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Integer id;
```

Strategy used by identity:

Insert a row without specifying a value for the ID. After inserting the row, ask the database for the last generated ID.

1.3 GenerationType.SEQUENCE

- GenerationType.SEQUENCE is the advised way to generate primary key values and hibernate uses a database sequence to generate unique values.

```
@Id
@GeneratedValue(strategy = GenerationType.SEQUENCE)
private Integer id;
```

- Strategy used by sequence:
- Before inserting a new row, ask the database for the next sequence value, then insert this row with the returned sequence value as ID.

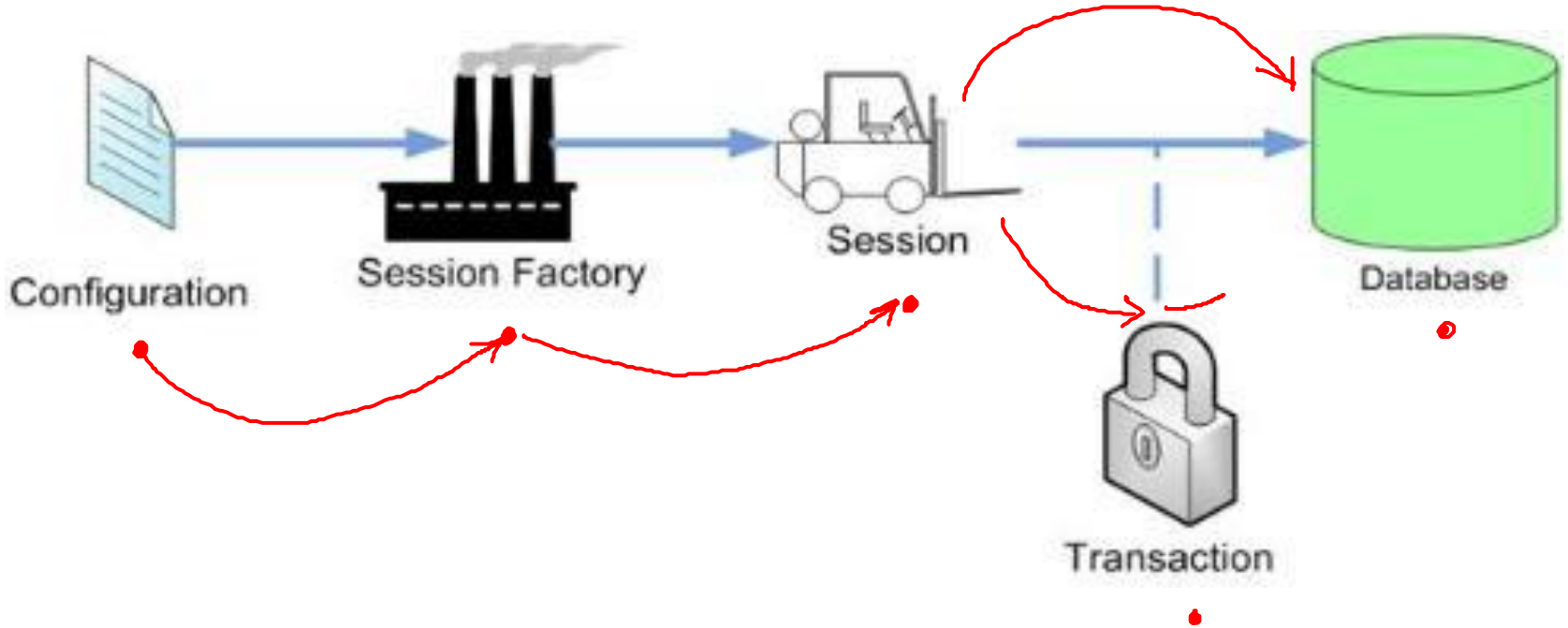
1.4 GenerationType.TABLE

- The GenerationType.TABLE gets only rarely used nowadays. It simulates a sequence by storing and updating its current value in a database table which requires the use of pessimistic locks which put all transactions into a sequential order. This slows down your application.

```
1.@Id
2.@GeneratedValue(strategy = GenerationType.TABLE)
3.private Long id;
```

main class

Working of hibernate



Configuration

- It represents a configuration or properties file for Hibernate. The Configuration object is usually created once during application initialization. The Configuration object reads and establishes the properties Hibernate uses to get connected to a database and configure itself for work. A Configuration object is used to create a SessionFactory and then typically is discarded.

SessionFactory

- The SessionFactory is created from a Configuration object, and as its name implies it is a factory for Session objects.The SessionFactory is an expensive object to create. It, like the Configuration object, is usually created during application start up. However, unlike the Configuration object, It should be created once and kept for later use.

- The SessionFactory object is used by all the threads of an application. It is a thread safe object. One SessionFactory object is created per database. Multiple SessionFactory objects (each requiring a separate configuration) are created when connecting to multiple databases. The SessionFactory can also provide caching of persistent objects.

Session

- Session objects provide the main interface to accomplish work with the database. Persistent objects are saved and retrieved through a Session object. A Session object is lightweight and inexpensive to create. A Session object does the work of getting a physical connection to the database. Session objects maintain a cache for a single application thread (request).

- Session objects are not thread safe. Therefore, session objects should not be kept open for a long time.Applications create and destroy these as needed. Typically, they are created to complete a single unit of work, but may span many units.

Transaction

- it represents unit of works.

Query and Criteria

- These objects are used to retrieve (and recreate) persistent objects.

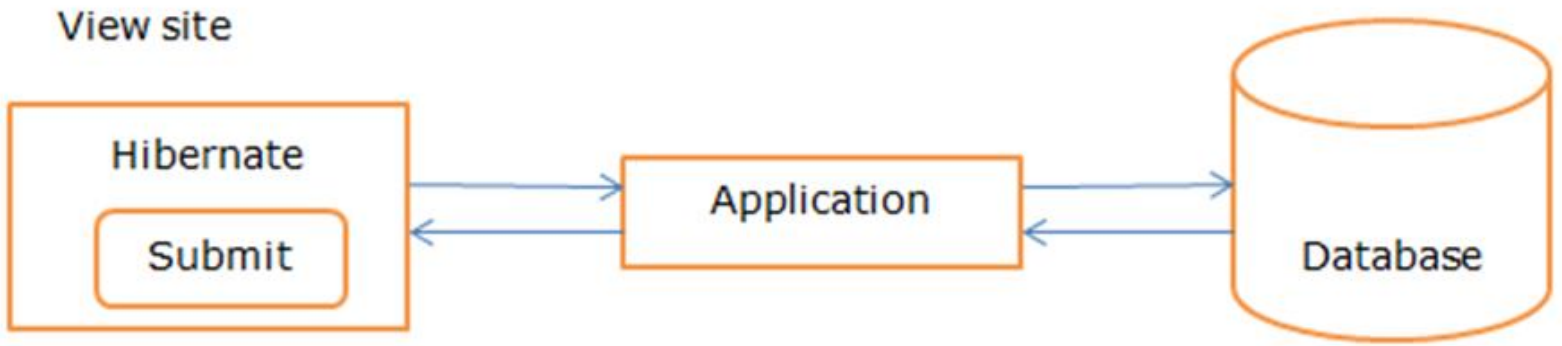
Hibernate Cache Support-

Cache memory
↓
RAM

- Caching is a mechanism to store the frequently retrieving data from DB into Cache Memory.
- The main advantage of using Cache is, it reduces the number of database calls and increases the performance of the application.
- Cache sits between application and database.

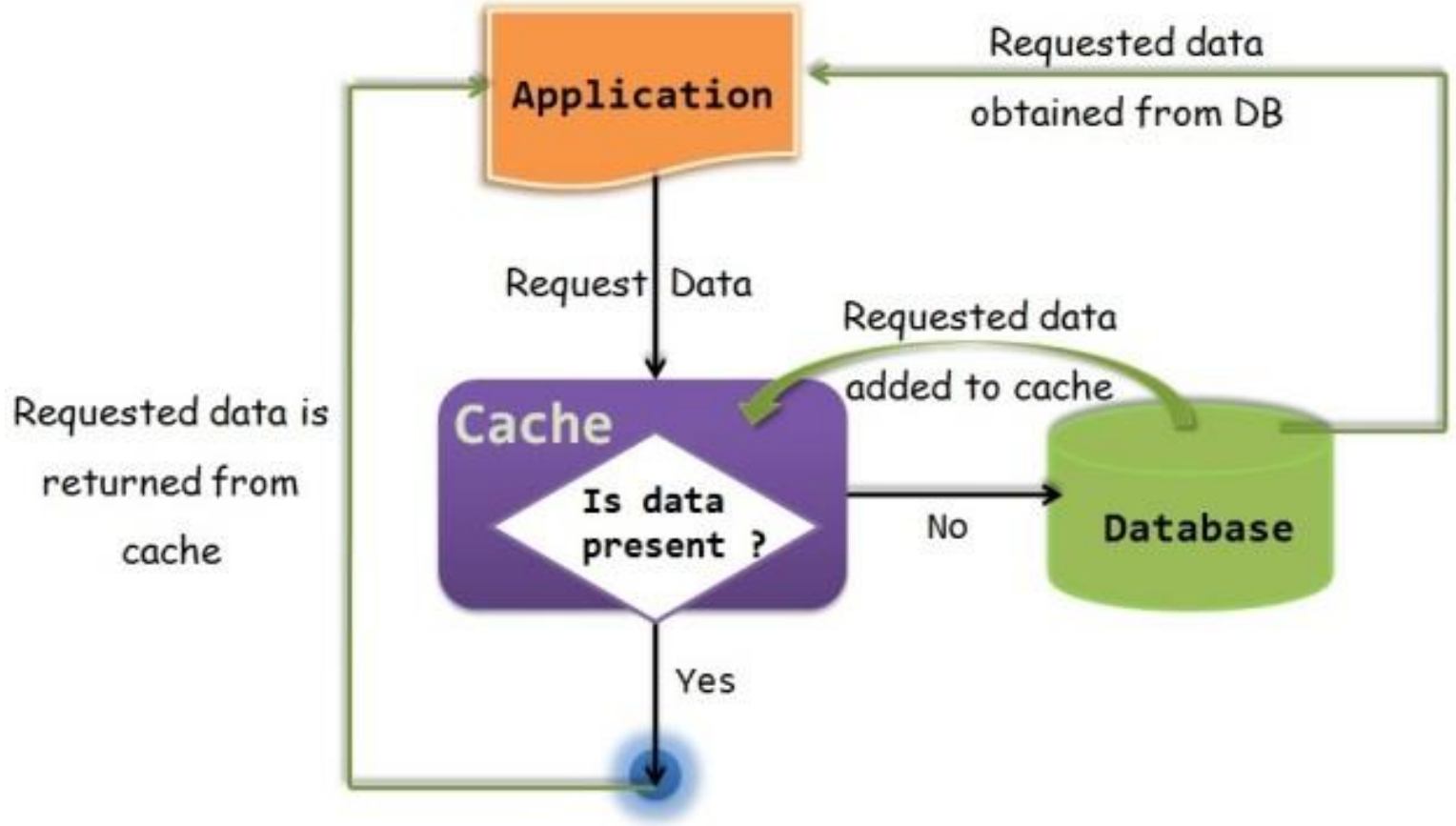
Problem without cache-

- Suppose I have Java study material application or site. It is the constant data, suppose 1 lakh user daily visit to site to read the concepts, so it will hit 1 lakh times to database due to this your application will slow and lot of processing time it will takes.
- To overcome this issue, we should go for cache supports.



How Application gets data from Cache ?

- Application first tries to find the data in cache and if requested data is not available in cache then only retrieve it from the DB and also put the same data in cache for later use.



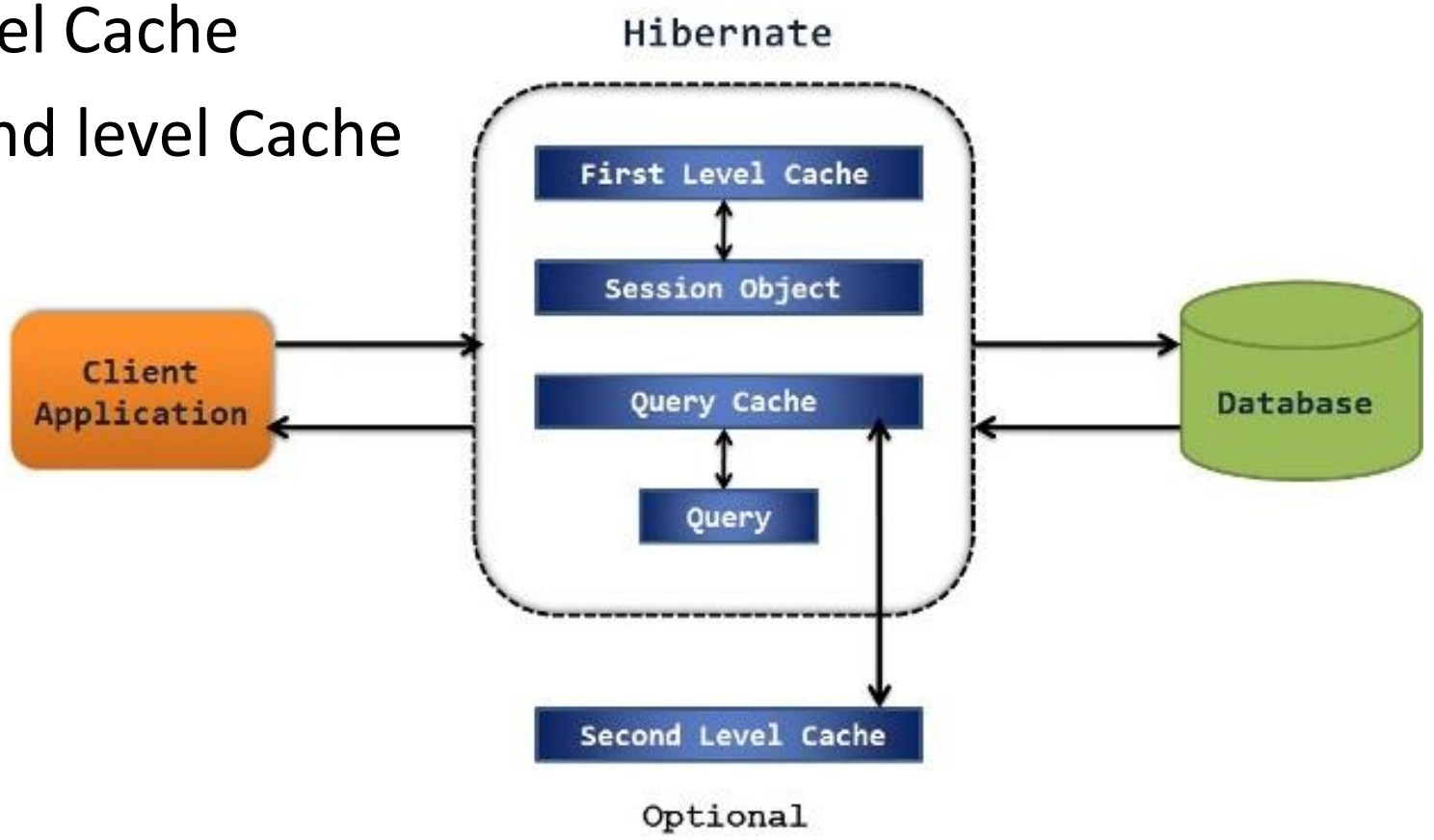
Cache in Hibernate

- One of the most powerful features of hibernate is Caching.
- Hibernate keeps the objects in the Cache memory to reduce the DB calls thereby increases the performance.
- Hibernate supports cache at different levels as explained below

1)Primary Cache or First level Cache

2)Secondary Cache or Second level Cache

3)Query Cache



Primary Cache

- First level cache also called Primary cache or Session level cache
- Primary cache is associated with the Session object, so all the objects within the hibernate session are kept in Primary cache to avoid multiple DB calls but once the session is closed all the cached objects will be lost.
- Primary cache is enabled by default and we don’t have any control to disable it.
- When to use?
- Example- Login to Gmail application, if you want to retrieve the inbox mails at first time login. It will load the data from database. If you trying to refresh and if you do not have new mails. The data instead of reading every time from database, it will load from cache itself until doing logout. It will do only one select operation.

- Note- It will fetch data only one time from database and store it on session objects next time when user request some data, so it will retrieve data from session objects.

Coding….

Query Cache

- Query cache will cache the results of the query against the object.
- If we have queries which runs multiple times with the same parameters then Query caching is best to use to avoid multiple DB calls.
- Query cache is disabled by default and we can enable it using configuration.
- We just need to set the hibernate.cache.use_query_cache property to true to enable Query cache.

Coding…

Secondary Cache

- Second level cache also called Secondary cache or SessionFactory level cache
- Secondary cache is associated with the SessionFactory and hence its available to the entire application.
- So objects kept in the secondary cache is available across multiple sessions.
- Secondary cache is disabled by default and we can enable it anytime using Configuration.

- There are various third party implementation providers for secondary cache and some of them are

- EH Cache
- JBoss Cache
- OS Cache
- Swarm Cache

Coding…

HibernateMavenProject/pom.xml - Eclipse IDE

File Edit Source Navigate Search Project Run Design Window Help

Client.java | HibernateMavenProject/pom.xml

```
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.30</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-ehcache</artifactId>
            <version>5.6.10.Final</version>
        </dependency>
        <dependency>
            <groupId>net.sf.ehcache</groupId>
            <artifactId>ehcache</artifactId>
            <version>2.10.6</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>2.0.3</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

Writable | Insert | 32 : 22 : 1123

Type here to search

ENG 20:17 30-09-2022

HibernateMavenProject/src/main/java/hibernate.cfg.xml - Eclipse IDE

File Edit Source Source Navigate Search Project Run Window Help

Client.java | HibernateMavenProject/pom.xml | hibernate.cfg.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE hibernate-configuration SYSTEM
3       "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
4  <hibernate-configuration>
5      <session-factory>
6          <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
7          <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/test</property>
8          <property name="hibernate.connection.username">root</property>
9          <property name="hibernate.connection.password">root</property>
10         <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
11         <property name="hbm2ddl.auto">update</property>
12         <property name="show_sql">true</property>
13         <property name="hibernate.cache.use_second_level_cache">true</property>
14         <property name="hibernate.cache.region.factory_class">org.hibernate.cache.ehcache.EhCacheRegionFactory</property>
15         <property name="hibernate.javax.cache.provider">net.sf.ehcache.hibernate.EhCacheProvider</property>
16         <mapping class="com.velocity.Employee" />
17     </session-factory>
18 </hibernate-configuration>
```

Design | Source

hibernate-configuration/session-factory/property/#text | Writable | Smart Insert | 15 : 88 : 969

Type here to search

ENG 20:20 30-09-2022

HibernateMavenProject/src/main/java/com/velocity/Employee.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Client.java | HibernateMavenProject/pom.xml | hibernate.cfg.xml | Employee.java

```
3  import javax.persistence.Column;
4  import javax.persistence.Entity;
5  import javax.persistence.GeneratedValue;
6  import javax.persistence.GenerationType;
7  import javax.persistence.Id;
8  import javax.persistence.Table;
9
10 import org.hibernate.annotations.Cache;
11 import org.hibernate.annotations.CacheConcurrencyStrategy;
12
13 @Entity
14 @Table (name ="emp")
15 @Cache(usage = CacheConcurrencyStrategy.READ_WRITE)
16 public class Employee {
17     @Id
18     @Column(name ="emp_id")
19     @GeneratedValue (strategy = GenerationType.AUTO)
20     private int id;
21     @Column (name = "emp_fullName")
22     private String fullName;
23     @Column (name = "emp_city")
24     private String city;
25     @Column (name = "emp_age")
26     private int age;
27     @Column (name ="emp_salary")
28     private long salary;
```

Writable | Smart Insert | 15 : 52 : 429

Type here to search

ENG 20:23 30-09-2022

DemoCache/src/main/java/com/velocity/in/Client.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- arrays
- bank
- CollectionFramework
- constructorchaining
- Demo
- DemoCache
  - src/main/java
    - com.velocity.in
      - Client.java
      - Employee.java
    - hibernate.cfg.xml
  - src/main/resources
  - src/test/java
  - src/test/resources
  - JRE System Library [J2SE-1.5]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- DemoHibernate
- Encapsulation
- Exceptions
- HibernateDemo
- HibernateMavenProject
- Inheritance
- inputoutput
- JSPDemo
- Jumpingstatement
- lamdaExpression
- multitasking
- Serialization
- Servers
- StringProject

Client.java | hibernate.cfg.xml | Employee.java | DemoCache/pom.xml

```
6
7  public class Client {
8
9      public static void main(String[] args) {
10         Configuration cfg = new Configuration();
11         cfg.configure("hibernate.cfg.xml");
12         SessionFactory sessionFactory = cfg.buildSessionFactory();
13         Session session = sessionFactory.openSession();
14         Employee emp = session.load(Employee.class, 1);
15         System.out.println(emp);
16         //Second session
17         Session session2 = sessionFactory.openSession();
18         Employee emp2 = session2.load(Employee.class, 1);
19         System.out.println(emp2);
20
21     }
```

Problems | Javadoc | Console | Progress

Client (1) [Java Application] C:\Program Files\Java\jdk1.8.0_331\bin\javaw.exe (30-Sep-2022, 9:11:05 pm) [pid: 17676]

```
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.intern
Sep 30, 2022 9:11:09 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator in
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.inte
Hibernate: select employee0_.emp_id as emp_id1_0_0_, employee0_.emp_age as emp_age2_0_0_, employee0_.e
Employee [id=1, fullName=Pawan patil, city=pune, age=30, salary=50000]
Employee [id=1, fullName=Pawan patil, city=pune, age=30, salary=50000]
```

Writable | Smart Insert | 16 : 25 : 501

Type here to search

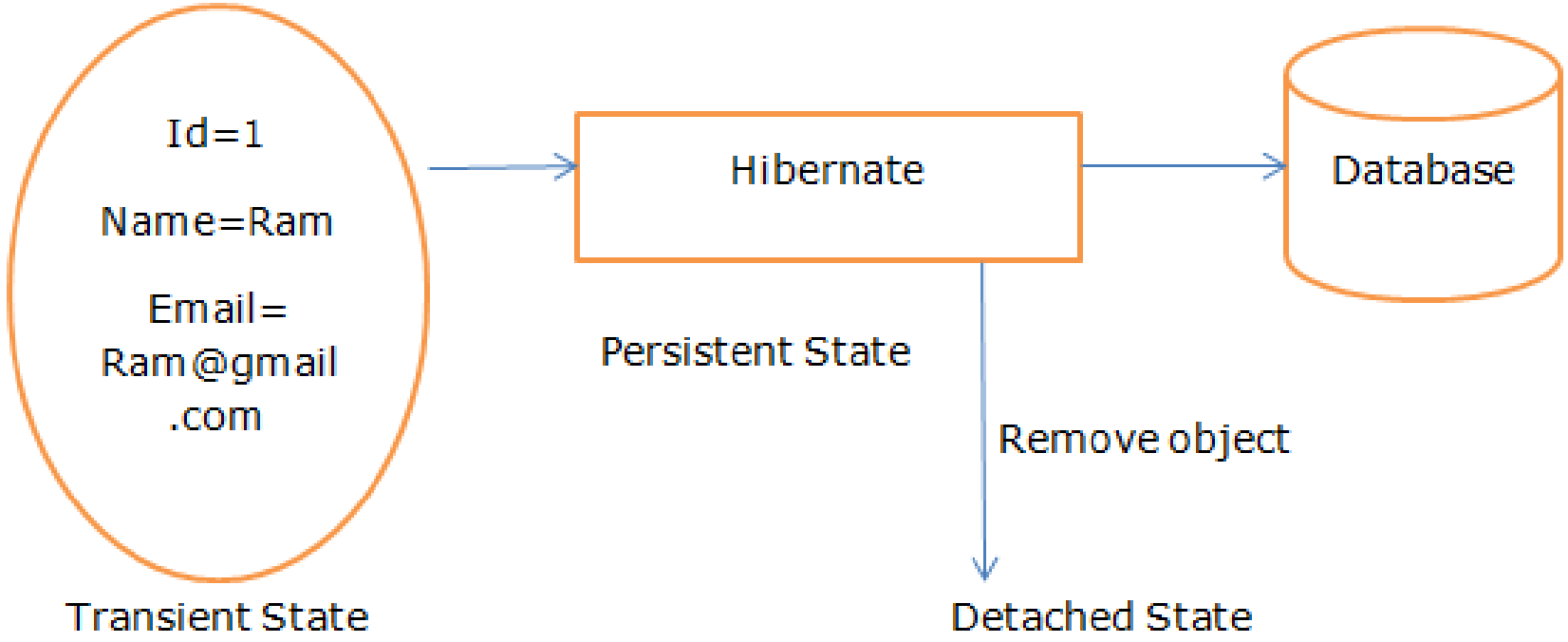
ENG 21:11 30-09-2022

Hibernate object states-

- It is called as life cycle of POJO class objects.
- There are three types of objects in the hibernate as-

- Transient state
- Persistent state
- Detached state

Hibernate object states-



1. Transient Object State:

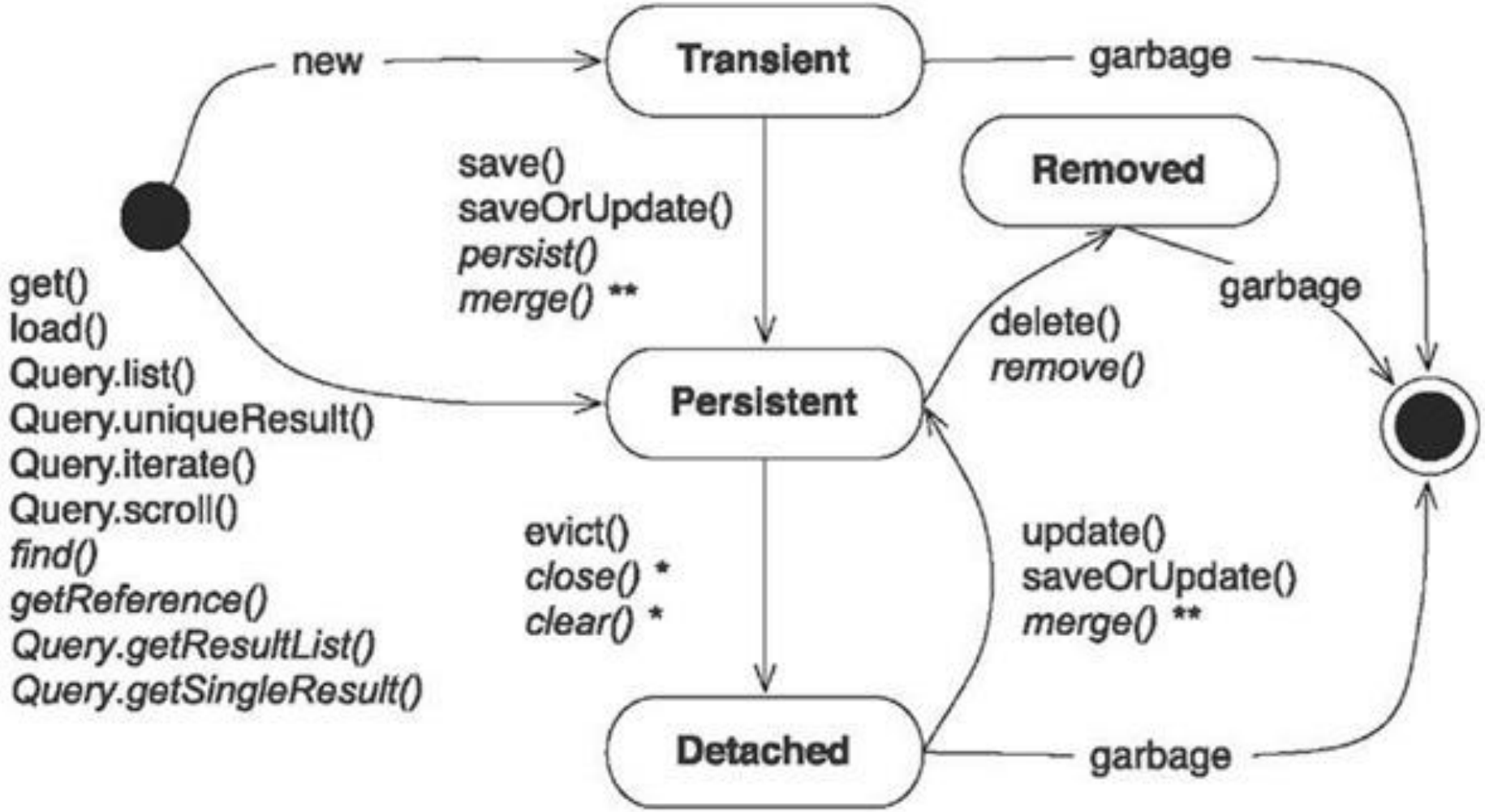
- An object which is not associated with hibernate session and does not represent a row in the database is considered as transient. It will be garbage collected if no other object refers to it.
- An object that is created for the first time using the new() operator is in transient state. When the object is in transient sate then it will not contain any identifier (primary key value). You have to use save, perist or saveOrUpdate methods to persist the transient object.
- Employee emp = new Employee();
- emp.setName(“Suresh Jadhav");

2. Persistent Object State:

- An object that is associated with the hibernate session is called as Persistent object. When the object is in persistent state, then it represent one row of the database and consists of an identifier value.You can make a transient instance persistent by associating it with a Session.
- Long id = (Long) session.save(emp);
- // emp object is now in a persistent state

3. Detached Object State:

- Object which is just removed from hibernate session is called as detached object.The sate of the detached object is called as detached state.
- When the object is in detached sate then it contain identity but you can’t do persistence operation with that identity.
- Any changes made to the detached objects are not saved to the database. The detached object can be reattached to the new session and save to the database using update, saveOrUpdate and merge methods.
- session.close();
- //object in detached state



* Hibernate & JPA, affects all instances in the persistence context
** Merging returns a persistent instance, original doesn't change state

Thank You