```
Interaction with more than 1 DBS/w using hibernate
===================================================
   UseCase:: copy one bank customer details to another bank(when banks get merged)
                  transfer money from one account to another account(both accounts
belongs to different bank)
                                      refer: HB-21-InteractionwithMulitpleDB



Working with LOB's
==================
=> Images,audiofile,videofiles are called BinaryLargeObjects(BLOB) because they
internally manage the data as binary information.
=> textfiles,rich text files and etc are called CharacterLargeObjects(CLOB) because
they internally manage the data in the form of
   character type of data.
=> BLOB (byte[] + @Lob)
=> CLOB (char[] + @Lob)

Code for reading byte data and character data
---------------------------------------------
      byte imageContent[]=null;
      char textContent[]=null;
      try(  FileInputStream fis=new FileInputStream("marriage.jpg")){

                  //prepare byte[] from image file
                  imageContent=new byte[fis.available()];
                  fis.read(imageContent);

                  //prepare char[] form text file
                  File file=new File("resume.txt");
                  try(FileReader reader=new FileReader(file)){
                        textContent=new char[(int) file.length()];
                        reader.read(textContent);
                  }//try2
      }//try1
            catch(IOException ioe) {
                  ioe.printStackTrace();
            }catch(Exception e) {
                  e.printStackTrace();
            }
      }

Code for Writing byte data and character data to a file
-------------------------------------------------------
//create Dest image file having byte[] imageContent
      fos=new FileOutputStream("store/photo.jpg");
      fos.write(seeker.getPhoto());

//Create a Dest resume.txt file having char[] textContent
      writer=new FileWriter("store/resume.txt");
      writer.write(seeker.getResume());
      fos.flush();
      writer.flush();

                        refer:: HB-22-HibernateLobOperation
```

```
Locking
-------
If multiple threads/app's simultaneously accessing and manipulating the records
then there is a possibility of getting
concurrency problem.
        To avoid this problem, we need to lock the record in hibernate.
        Locking can be done in 2 ways
                a. optimistic locking
                                => Allows second thread/apps simultaneously to access
and modify the record, first app notices the modification
                                    and throws Exception
                                => To use this feature we need to enable @Version
annotation in entity class.

                b. pesimistic locking
                                => First thread/app locks the record,so if the second
thread tries to access and modify the record then it
                                    would result in "Exception".
                                => To use this feature we need to use
session.get(,,LockMode.UPGRADE_NOWAIT) as the third argument value.


Optimistic Lock
===============
client1.java
            After getting the record, make the thread to sleep for 30seconds
client2.java
            Get the same record and make some changes.

                        refer:: HB-23-HibernateOptimisticLocking


Pesimistic Lock
===============
client1.java
            After getting the record(use session.get(,,LockMode.UPGRADE_NOWAIT),
make the thread to sleep for 30seconds
client2.java
            Get the same record(use session.get(,,LockMode.UPGRADE_NOWAIT) and make
some changes.

                        refer:: HB-24-HibernatePesimiticLocking


Bulk operation in hibernate
---------------------------
To select or mainpulate one/more record or object having our choice as criteria
value, we need to go for this bulk operation concept
        a. HQL/JPQL
        b. Native SQL
        c. Criteria API

Note: JPQL=> It is a specification given by SUNMS which speaks about the rules to
develop object based query Language
        HQL=> It is an implementation of JPQL by hibernate.

1. HQL [Hibernate Query Language]
=> HQL is a pwerfull query language provided by Hibernate inorder to perform
manipulations over multiple records.
```

=> HQL is an object oriented query language, it able to support for the object oriented features like
    encapsulation,polymorphism,...., but, SQL is structered query language.
=> HQL is database independent query language, but, SQL is database dependent query language.
=> In case of HQL, we will prepare queries by using POJO class names and their properties, but, in case of SQL, we will prepare
    queries on the basis of database table names and table columns.
=> HQL queries are prepare by using the syntaxes which are similar to SQL queries syntaxes.
=> HQL is mainly for retrival operations , but, right from Hibernate3.x version we can use HQL to perform insert , update and
    delete operations along with select operations, but, SQL is able to allow any type of database operation.
=> In case of JDBC, in case of SQL, if we execute select sql query then records are retrived from database table and these
    records are stored in the form of ResultSet object, which is not implementing java.io.Serializable , so that, it is not possible
    to transfer in the network,but, in the case of HQL, if we retrive records then that records will be stored in Collection objects,
    which are Serializable bydefault, so that, we are able to carry these objects in the network.
=> HQL is database independent query language, but, SQL is database dependent query language.
=> In case of Hibernate applications, if we process any HQL query then Hibernate Software will convert that HQL Query into database
    dependent SQL Query and Hibernate software will execute that generated SQL query.

Note: HQL is not suitable where we want to execute Database dependent sql queries
EX: PL/SQL procedures and functions are totally database dependent, where we are unable to use HQl queries.

Note:

eg: SQL> SELECT * FROM EMP WHERE EMPNO>? AND EMPNO<?
    HQL> FROM in.ineuron.model.Employee WHERE eno>? AND eno<?

    SQL> DELETE FROM EMP WHERE EMPNO=?
    HQL> DELETE FROM in.ineuron.model.Employee WHERE eno=?

    SQL>SELECT ENO,ENAME FROM EMPLOYEE
    HQL>SELECT eno,ename FROM in.ineuron.model.Employee

    SQL>UPDATE EMPLOYEE SET ENAME=?,ESAL=? WHERE ENO=?
    HQL>UPDATE in.ineuron.model.Employee SET ename=?,esal=? WHERE eno=?

                        refer:: HB-25-HQLAPP