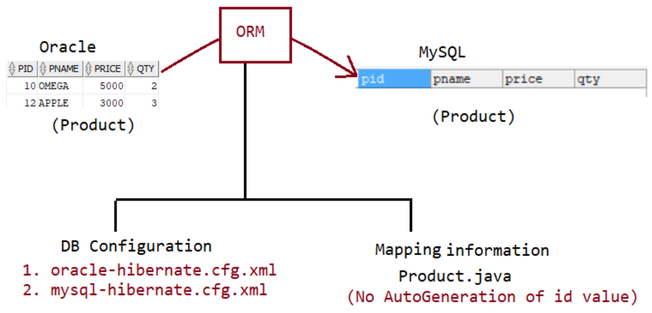
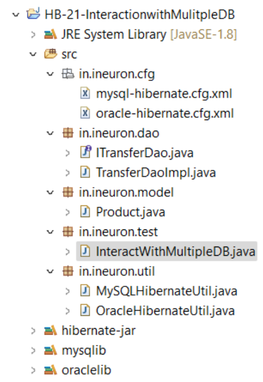
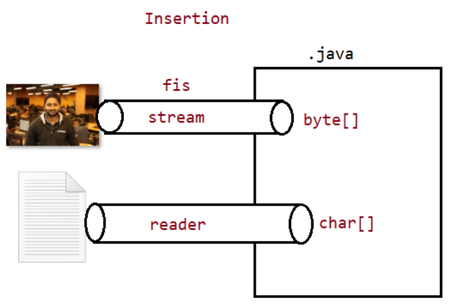
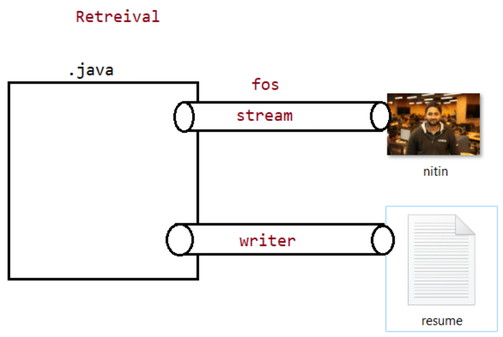
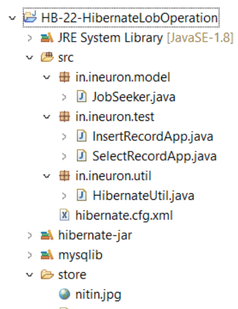
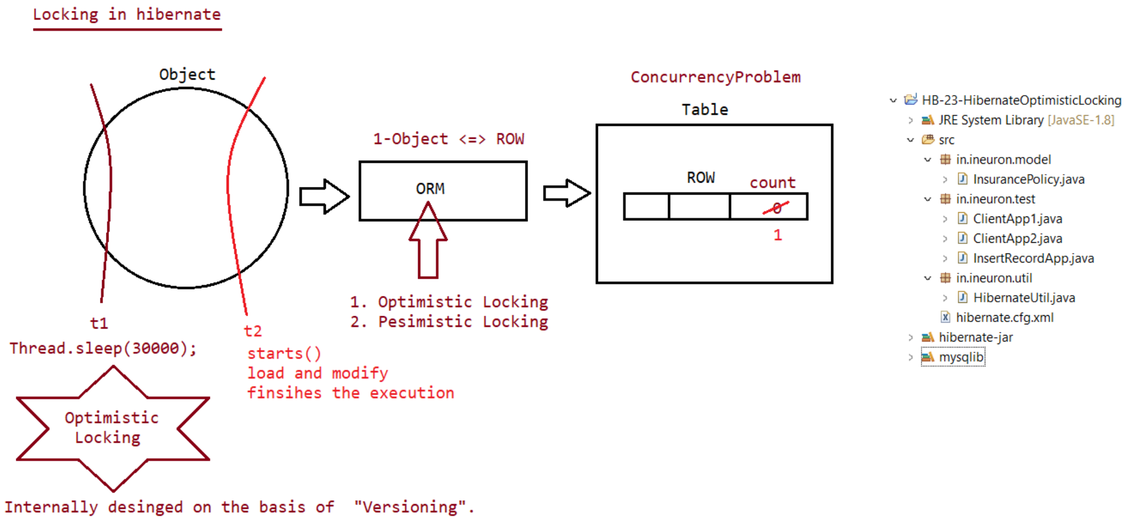

Oracle
PID PNAME PRICE QTY
10 OMEGA 5000 2
12 APPLE 3000 3
(Product)
ORM
MySQL
pid pname price qty
(Product)
DB Configuration
1. oracle-hibernate.cfg.xml
2. mysql-hibernate.cfg.xml
Mapping information
Product.java
(No AutoGeneration of id value)


HB-21-InteractionwithMulitpleDB
JRE System Library [JavaSE-1.8]
src
in.ineuron.cfg
mysql-hibernate.cfg.xml
oracle-hibernate.cfg.xml
in.ineuron.dao
ITransferDao.java
TransferDaoImpl.java
in.ineuron.model
Product.java
in.ineuron.test
InteractWithMultipleDB.java
in.ineuron.util
MySQLHibernateUtil.java
OracleHibernateUtil.java
hibernate-jar
mysqlib
oraclelib

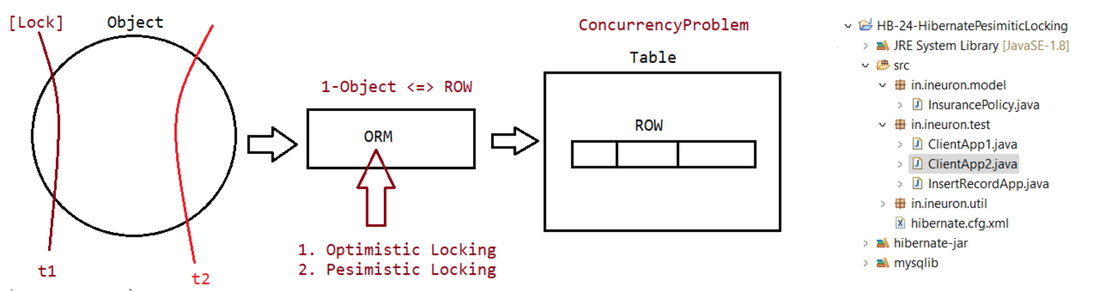

Insertion
.java
fis
stream
byte[]
reader
char[]


Retreival
.java
fos
stream
nitin
writer
resume

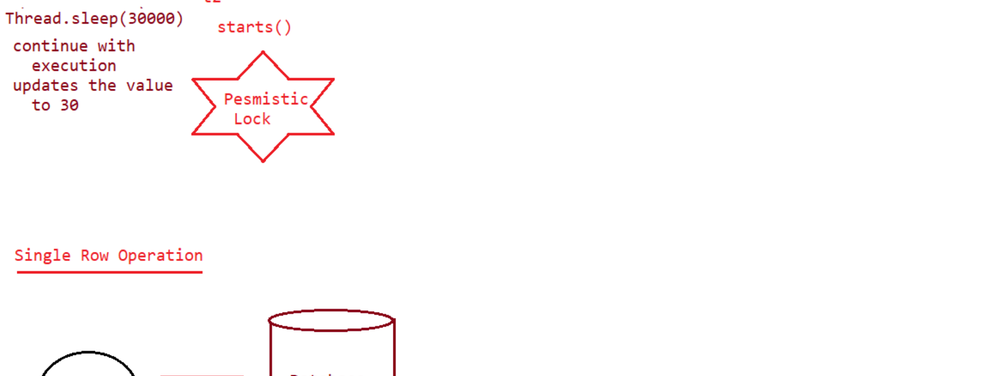

HB-22-HibernateLobOperation
JRE System Library [JavaSE-1.8]
src
in.ineuron.model
JobSeeker.java
in.ineuron.test
InsertRecordApp.java
SelectRecordApp.java
in.ineuron.util
HibernateUtil.java
hibernate.cfg.xml
hibernate-jar
mysqlib
store
nitin.jpg
resume.txt

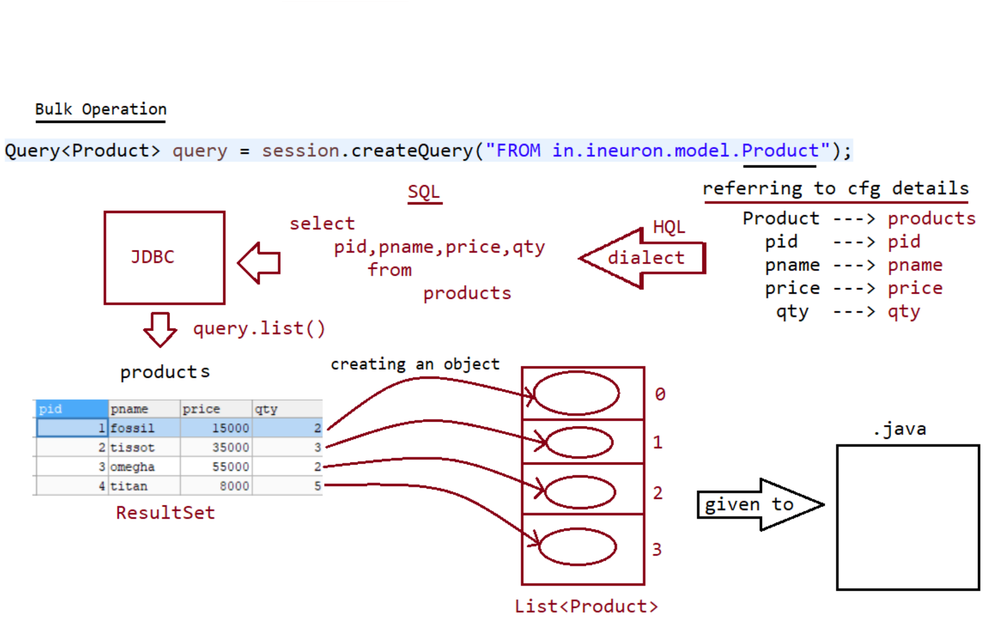

Locking in hibernate
Object
t1
t2
Thread.sleep(30000);
1-Object <=> ROW
ORM
1. Optimistic Locking
2. Pesimistic Locking
ConcurrencyProblem
Table
ROW
count
1
starts()
load and modify
finsihes the execution
Optimistic Locking
Internally desinged on the basis of "Versioning".

HB-23-HibernateOptimisticLocking
JRE System Library [JavaSE-1.8]
src
in.ineuron.model
InsurancePolicy.java
in.ineuron.test
ClientApp1.java
ClientApp2.java
InsertRecordApp.java
in.ineuron.util
HibernateUtil.java
hibernate.cfg.xml
hibernate-jar
mysqlib


PesimisticLock
[Lock]
Object
t1
t2
Thread.sleep(30000)
continue with
execution
updates the value
to 30
1-Object <=> ROW
ORM
1. Optimistic Locking
2. Pesimistic Locking
ConcurrencyProblem
Table
ROW
starts()
Pesmistic Lock

HB-24-HibernatePesimiticLocking
JRE System Library [JavaSE-1.8]
src
in.ineuron.model
InsurancePolicy.java
in.ineuron.test
ClientApp1.java
ClientApp2.java
InsertRecordApp.java
in.ineuron.util
hibernate.cfg.xml
hibernate-jar
mysqlib


Single Row Operation
Object
id
ORM
Database
(SQL)
pkvalue
SRS to inform the orm tools to implement bulk operation
JPQL
HQL
EQL
IQL
Query will not be written rather query will be mapped to
a. tablename ====> entity name
b. columname ====> field name/property name
Bulk Operation
HQL
ORM
Database
(SQL)


Bulk Operation
Query<Product> query = session.createQuery("FROM in.ineuron.model.Product");
SQL
select
pid,pname,price,qty
from
products
HQL
dialect
referring to cfg details
Product ---> products
pid ---> pid
pname ---> pname
price ---> price
qty ---> qty
JDBC
query.list()
products
pid pname price qty
1 fossil 15000 2
2 tissot 35000 3
3 omegha 55000 2
4 titan 8000 5
ResultSet
creating an object
0
1
2
3
List<Product>
given to
.java

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