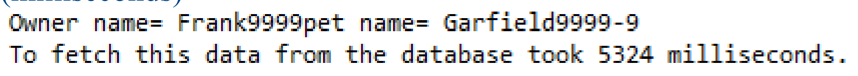
Here is the experiment results

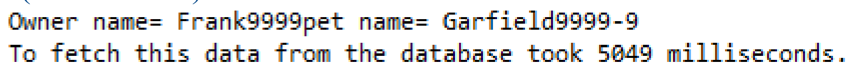
a) Run the application as is, without any optimization, and note the time. Run the application 3 times to get an average time (speed ups and slowdowns can be caused by background processes and / or caching).

1st try took : 5324 (milliseconds)



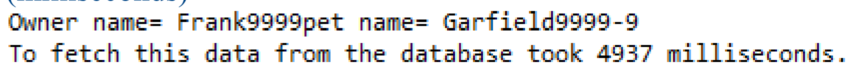
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 5324 milliseconds.
```

2nd try took : 5049 (milliseconds)



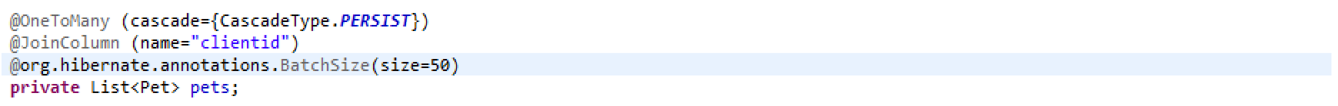
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 5049 milliseconds.
```

3rd try took: 4937 (milliseconds)



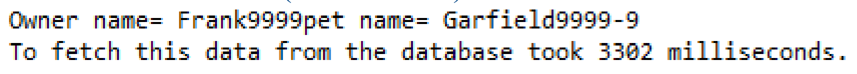
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 4937 milliseconds.
```

b) Modify the mapping for Owner.java to use batch fetching, batch size 10, run the application three times again and note the new time. Also check the time when using batch size 5 and when using batch size 50.



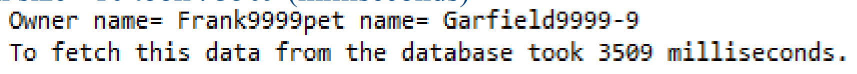
```
@OneToMany (cascade={CascadeType.PERSIST})
@JoinColumn (name="clientid")
@org.hibernate.annotations.BatchSize(size=50)
private List<Pet> pets;
```

1st try with batch size =10 took : 3302 (milliseconds)



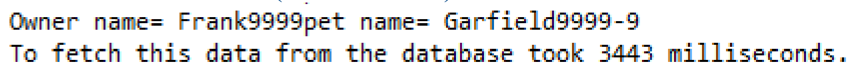
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 3302 milliseconds.
```

2nd try with batch size =10 took : 3509 (milliseconds)



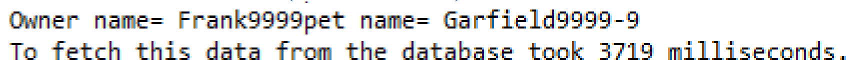
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 3509 milliseconds.
```

3rd try with batch size =10 took : 3443 (milliseconds)



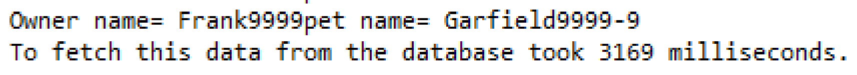
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 3443 milliseconds.
```

1st try with batch size =5 took : 3719 (milliseconds)



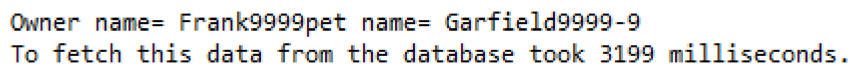
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 3719 milliseconds.
```

1st try with batch size =50 took : 3169 (milliseconds)



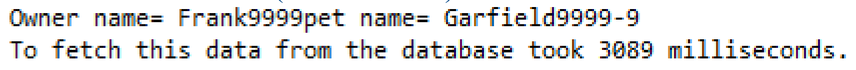
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 3169 milliseconds.
```

2nd try with batch size =50 took : 3199 (milliseconds)



```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 3199 milliseconds.
```

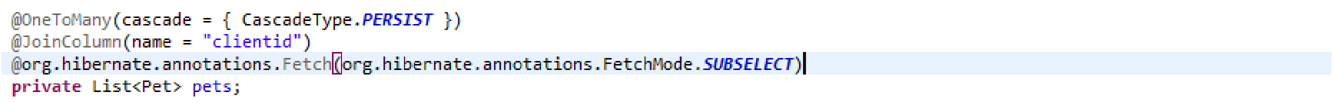
3rd try with batch size =50 took : 3089 (milliseconds)



```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 3089 milliseconds.
```

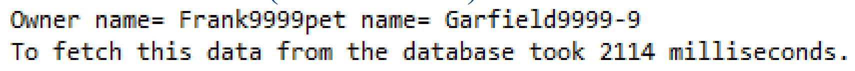
Conclusion: with batch size it improves performance significantly, with bigger batch size performance increased more compare to smaller size

c) Modify the mapping to use sub-select strategy instead of batch fetching, and run the application 3 times to note the time it took to retrieve the results.



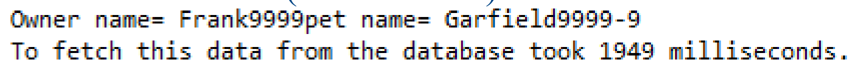
```
@OneToMany(cascade = { CascadeType.PERSIST })
@JoinColumn(name = "clientid")
@org.hibernate.annotations.Fetch(org.hibernate.annotations.FetchMode.SUBSELECT)
private List<Pet> pets;
```

1st try with subselect took : 2114 (milliseconds)



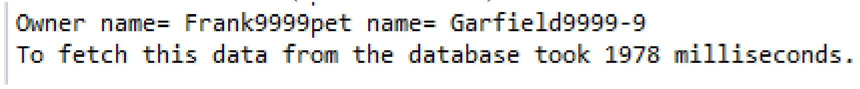
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 2114 milliseconds.
```

2nd try with subselect took : 1949 (milliseconds)



```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 1949 milliseconds.
```

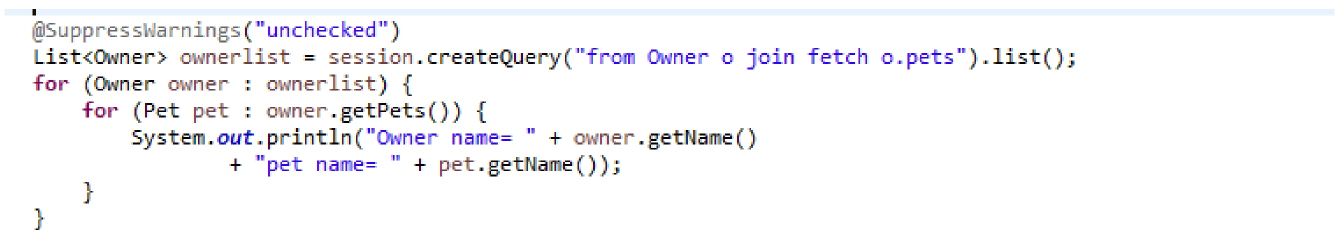
3rd try with subselect took : 1978 (milliseconds)



```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 1978 milliseconds.
```

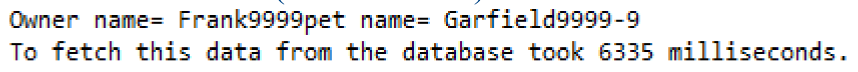
Conclusion: its even better than batch size, performance improved significant

d) Remove the sub-select strategy and use a join fetch query in Application.java to retrieve everything, again running the application 3 times and checking the time. Also check the difference between using a named query, or just a query directly in code.



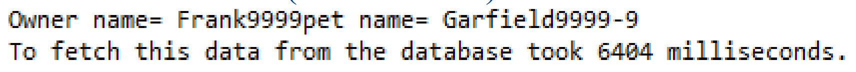
```
@SuppressWarnings("unchecked")
List<Owner> ownerlist = session.createQuery("from Owner o join fetch o.pets").list();
for (Owner owner : ownerlist) {
    for (Pet pet : owner.getPets()) {
        System.out.println("Owner name= " + owner.getName()
                + "pet name= " + pet.getName());
    }
}
```

1st try with join fetch took : 6335 (milliseconds)



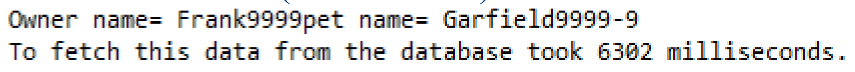
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 6335 milliseconds.
```

2nd try with join fetch took : 6404 (milliseconds)



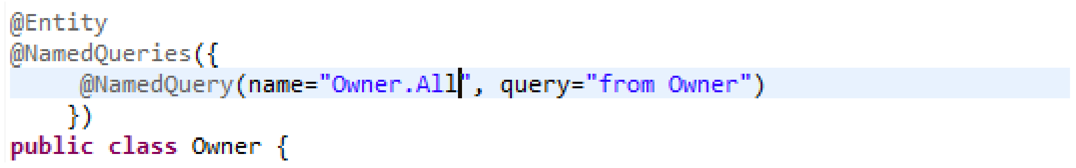
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 6404 milliseconds.
```

3rd try with join fetch took : 6302 (milliseconds)

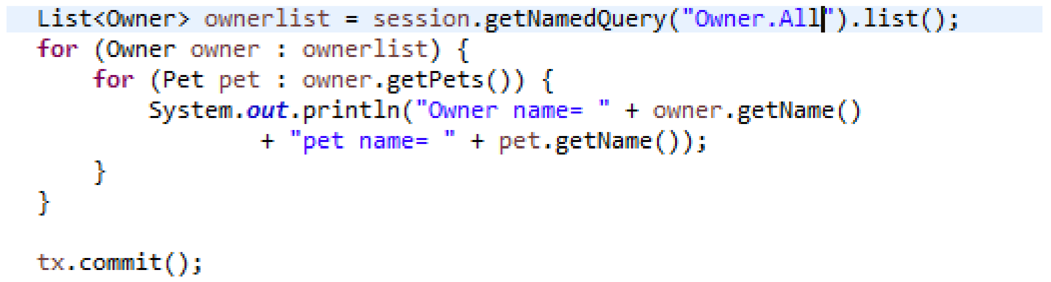


```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 6302 milliseconds.
```

With named Query:

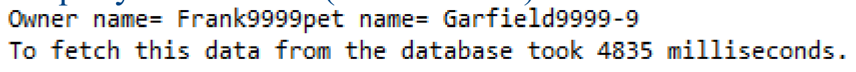


```
@Entity
@NamedQueries({
    @NamedQuery(name="Owner.All", query="from Owner")
   })
public class Owner {
```



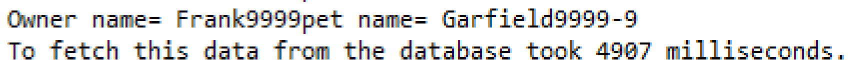
```
  List<Owner> ownerlist = session.getNamedQuery("Owner.All").list();
  for (Owner owner : ownerlist) {
      for (Pet pet : owner.getPets()) {
          System.out.println("Owner name= " + owner.getName()
                  + "pet name= " + pet.getName());
      }
  }

  tx.commit();
```

1st try with named query took: 4835 (milliseconds)



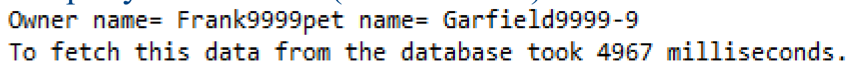
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 4835 milliseconds.
```

2nd try with named query took: 4907 (milliseconds)



```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 4907 milliseconds.
```

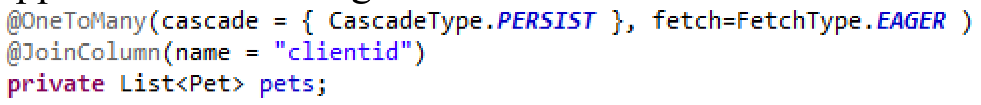
3rd try with named query took: 4967 (milliseconds)



```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 4967 milliseconds.
```

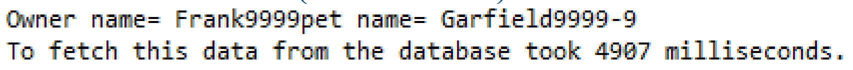
Conclusion: Named query gives better performance than join fetch, interesting.....

e) Lastly modify the application to use the FetchType.EAGER strategy. Run the application 3 times again and note the time.



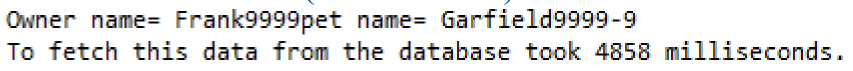
```
@OneToMany(cascade = { CascadeType.PERSIST }, fetch=FetchType.EAGER )
@JoinColumn(name = "clientid")
private List<Pet> pets;
```

1st try with eager fetch took: 4907 (milliseconds)



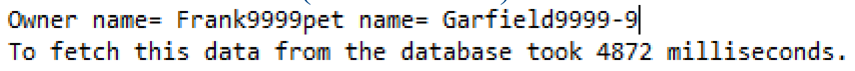
```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 4907 milliseconds.
```

2nd try with eager fetch took: 4858 (milliseconds)



```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 4858 milliseconds.
```

3rd try with eager fetch took: 4872 (milliseconds)



```
Owner name= Frank9999pet name= Garfield9999-9
To fetch this data from the database took 4872 milliseconds.
```

Conclusion: doesn't show any big difference compare to named query