Hibernate Optimization (HOP)

Exercise HOP.1 – Data Fetching

The Setup:

In this exercise we will use System.nanotime() to check how long it takes for MySQL and Hibernate to retrieve the same dataset with different fetching strategies.

Start this exercise by downloading the project from **Lab7Optimization** from Sakai and add the Hibernate dependencies to the pom.xml.

The Application:

The application has a Populate.java file that will insert 100,000 owner objects, each with 10 associated pet objects into the database. Run it once (will take a while).

Then change line 24 of the persistence.xml file to have the value of “none” instead of “drop-and-create”. This will stop the tables from being re-created every time and keeping you from having to recreate all the data.

Then run App.java, which will create an N+1 and tell you how long it took.

The Exercise:

Consider what the application does, and write down which strategy you think will perform best under these circumstances. To get a more accurate time you should probably run each test 3 times and take the average, but once is okay to get an idea.

a) Add the @LazyCollection with option EXTRA to the association and run App again.

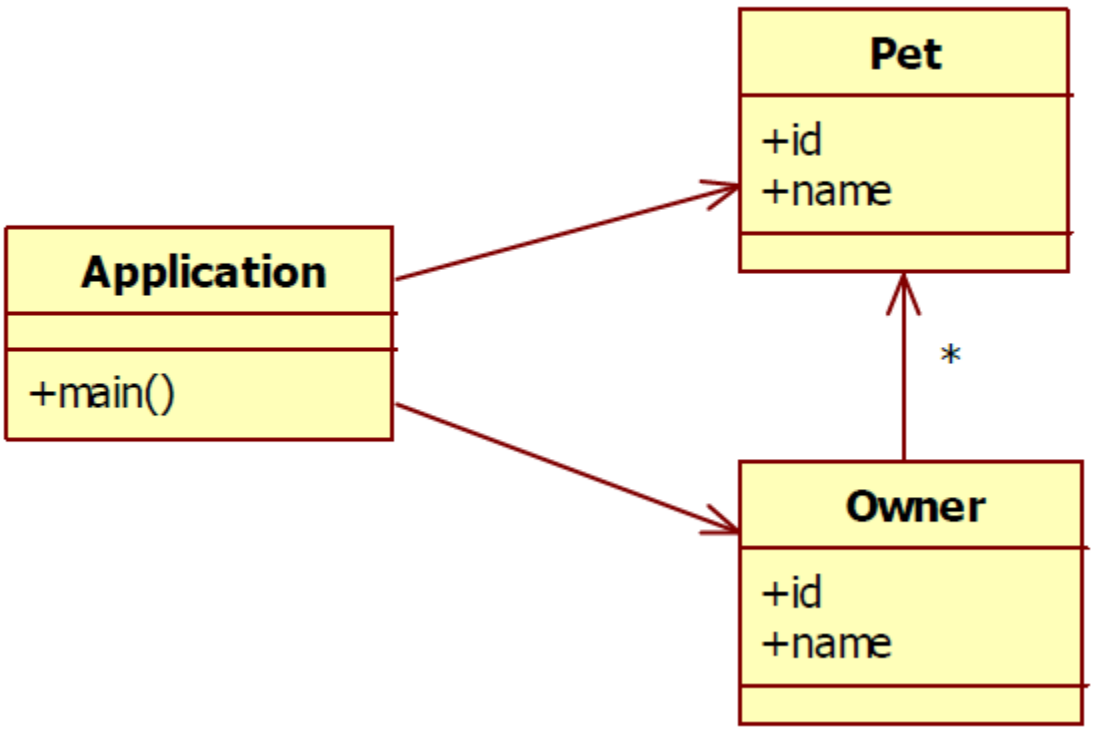
b) Remove the @LazyCollection, and modify the mapping for **Owner.java** to use **batch fetching**, batch size 10. Also check the time when using sizes 5 and 50.

c) Modify the mapping to use the **sub-select** strategy instead of batch fetching.

d) Remove the sub-select strategy and use a **join fetch query** in App.java to retrieve everything. Also check the difference between using a named query, or just a query directly in code.

e) Lastly modify the application to use an Entity Graph instead of a join fetch.

Check to see if the strategy you thought would perform best was indeed the best for this situation. Remember, just because a strategy performed well under these circumstances does not necessarily mean it will perform well under other circumstances.


Pet
+id
+name
Application
+main()
*
Owner
+id
+name