The Hibernate Framework Query Mechanisms Comparison

Tisinee Surapunt and Chartchai Doungsa-Ard

Abstract—The Hibernate Framework is an Object/Relational Mapping technique which can handle the data for database applications with the object oriented program. There is metadata to describe mapping relationships of persistence objects and database entities. The developers’ responsibilities take care only how programs work. Then the execution time of querying the data does not distinguish while the program is working. Therefore, the execution time of querying the data effects to overall execution time of the program. So, the author concerns about the application performance when queries the data. From the study, the execution time of querying the data is appropriate with difference query mechanisms and data environment. Not only the selected approach affects to the execution time, but other impact factors are also related. The factors consist of a number of data in the database, condition in the command and the number of records set. So, the author focuses on the testing query mechanism of a difference environment and conditions.

Index Terms—Hibernate framework, query mechanisms, performance query testing.

I. INTRODUCTION

Database Applications are widely used in many businesses. The valuable reports can be queried to support the business to make the correct decision such as a monthly report, annual report, or required up to date information. The applications connect to the DBMS using the developers’ favorite programming language. One of the most popular programming concepts is Object Oriented Programming (OOP). The OOP concept organizes the application by creating objects to hold attribute values and send messages among the objects. The benefits of the Object Oriented technique are reusable, easy to understand and reduced redundancy. While the developers turn entity stored in DBMS to objects in application, there may be an impedance mismatch occurs. The impedance mismatch is a technical problem based on the transformation between relational database and objects. This problem may reduce an application efficient of mapping objects with relational database [1]. The many-to-many relationship is an example of impedance mismatch problem. The relationship requires association tables, which are generated during the normalization process. Thus, the class diagram for the application is different from the normalized ER diagram, as a consequence, the mapping between the object, and the database entity object cannot be mapped directly. To solve the problem, ORM framework is proposed. The ORM framework [2] is the automatic tool which maps the structures of the classes in the application, and the tables in DMBS automatically. There are several ORM frameworks such as myBatis [3], JPA [4], [5] and Hibernate [6]. Among all, Hibernate framework is used widely because of its data query facilities. There are 3 query methods for the Hibernate framework which are Hibernate Query Language (HQL), Criteria API and Query for Native SQL. These query techniques are used to reduce the complexity of SQL commands and provide the object liked query. The key of using the Hibernate in the application is not only its convenient, but also the effectiveness. There are several key factors which relate to the performance of the Hibernate application such as the amount of query records and the cache technology [7].

In this study, the other factors, which may affect the query time, are investigated. The different Hibernate query mechanisms are focused in the different scenario. In Section II, the works related with the Hibernate performance analysis are discussed. The experimental design is described in Section III. The preliminary result and discussion are presented in Section IV. Finally, Section V is the conclusion and future work.

II. LITERATURE REVIEWS

A. Hibernate Framework

Hibernate Framework [6] is an Object/Relational Mapping technique which manages database tables with persistence objects automatically. The metadata is generated in order to describe mapping relationship between tables and objects. The fundamental feature is mapping database tables with persistence objects for object oriented programming. The architecture of Hibernate Framework is divided into 3 layers which are Presentation layer, Business Logic layer and Database Access layer as shown in Fig. 1.

![Diagram of Hibernate Framework](image)

Fig. 1. The architecture of hibernate framework [8].

Manuscript received January 10, 2014; revised March 20, 2014.

Tisinee Surapunt and Chartchai Doungsa-Ard are with Software Engineering Department, College of Arts Media and Technology, Chiang Mai University, Thailand (e-mail: tsurapunt@gmail.com, c.doungsaard@gmail.com).

DOI: 10.7763/LNSE.2014.V2.135
The Hibernate Framework is presented in the business logic tier. The programmers can customize the framework configuration and the object mapping to suit with the application. The Hibernate configuration is a crucial XML file to describe the configuration of database connection and project detailed. The file is usually named as hibernate.cfg.xml. The object mapping is the part which defines the relationship between the class design and the database design. The contents inside mapping file describe business rules of the application and mapping relationship between objects and database. There are 2 techniques to map database with persistence objects, which are XML mapping and Annotation mapping. For the XML mapping, the mapping information is prepared in the the XML file using the Hibernate dtd format, while the annotation mapping provides the mapping information in the source code directly [9]. The advantage of developing the application with the Hibernate Framework is reducing responsibilities and time consuming to develop an application with SQL commands. Furthermore, Hibernate Framework can handle with SQL commands when an application deals with persistence objects and the database. Hibernate Framework provides query mechanisms for retrieving data from the database, which are HQL (Hibernate Query Language), Criteria API and Name Query for Native SQL [10].

1) HQL (Hibernate query language)

HQL is a powerful query language which has syntax similar with SQL commands. Thus, the HQL commands are easy to understand for the legacy database application developers. The HQL approach queries data by using a class name and attributes instead of table name and columns. The example command of HQL command is shown in Fig. 2.

```
Query query = session.createQuery
        ("from BachelorStudent bachelorstu");
```

Fig. 2. An example of HQL commands.

The query in Fig. 2 will return all the BachelorStudent objects in the database and store in the query object.

2) Criteria API

Criteria API is focused on the Object Oriented technique instead of the query command. The query can be conducted by using the method calling from the session object. With the complex query, many method calls are invoked to get the result. This query technique is suited for the developers who have a background in object oriented technique. The example of Criteria API command is shown in Fig. 3.

```
Criteria cr = session.CreateCriteria
        ("BachelorStudent.class");
```

Fig. 3. An example of criteria API command.

The query in Fig. 3 also return all the BachelorStudent objects in the database and store in the cr object similarly to the query in Fig. 2. It is possible to get the same result using the different technique. So, we can measure for the performance.

3) Name query for native SQL

Name Query for Native SQL works with SQL commands mainly. This approach does not use the ability of Hibernate Framework. The developers can use SQL commands directly to query data, which they want.

B. Performance of Hibernate Framework

Currently, the performance of Hibernate Framework is measured by the query time [7], [11]. The researchers Wu, Hu, and Wang [11] studied on the impacts which influences the application performance. The significant impacts are separated into 3 groups, which are Cache Technology, Connection Pool Technology and Session Management Mechanism. From the 3 substantial impacts which can affect to query mechanism, there is an experiment to measure the query efficiency by the execution time. The experiment is designed to measure the query time before and after performance optimization. The average time before optimization is 375.2 ms where the average time after optimization spent 86.2 ms. The results show the significant time decreasing. The result shows that with proper optimization, the Hibernate framework can provide the better performance.

The effect of the secondary cache is also examined [7]. An experiment was designed to test the secondary cache performance in data persistence layer. The study focused on the query efficiency which selected Criteria API method. The authors queried data from 100 records and then increased the query data to 2,000 records by adding the query size for 500 records in each execution. The result showed that in the small query record size, the query efficiency is appropriate, while increasing the query size, the efficiency of the query time is not improved significantly.

III. THE COMPARISON OF QUERY MECHANISMS

In this research, the comparison performance between HQL and Criteria API methods are focused. The HQL is proposed for the programmers who are familiar with the SQL while the Criteria API is provided for the programmers who get used to the method calls. As the two mechanisms are created based on the users, the performance of each mechanism has not revealed yet. The experiment is designed to evaluate the query time for each mechanism. The HQL statements and the Criteria calls for query the same output are provided. Both of statements will query the data from the same the same data source with the same condition. The query time will be recorded. In order to compare the effectiveness of difference query mechanism operation between HQL and Criteria API approach, the experiment runs on a sample database. The BachelorStudent class is converted from a sample database which xxx is used for the experiment. The BachelorStudent class contained bachelor students’ information which collects 5 fields; student id, first name, last name, major id and GPA. The BachelorStudent class used in the experiment is shown in Fig. 4.

The experiment is performed on MySQL database with the difference amount of data in the database. In the experiment is defined an amount of data in the database at 30,000 and 100,000 records. All data have been generated randomly. The test is executed for 100 times per iteration. The experiment will be run for 20 iteration. The queried records set is restricted at 30,000 records. Because of the graph
representation will show the clearly different. The experiment has been divided into 2 part, the execution of small and large queried records set. The small queried records set started at 100 records until 1,000 records by increasing 100 records in each data size. There are 2 types of query, the query without criteria, and with criteria. The query without criteria is the query without any WHERE cause. The first query objects are queried using both techniques. The query statement for HQL query is shown in Fig. 2, and the statement for Criteria API is shown in Fig. 3. The query with criteria using the WHERE cause to select the top objects which pass the criteria. The gpa is used as the criteria. The pseudo code for generating the criteria is provided in Fig. 5.

The condition of WHERE clause is separated into 3 conditions. There are a greater than, a less than and an equal. After that, the experiment generated 2 random numbers for creating a condition in each execution. The first number is an integer for choosing condition to compare. The second number is GPA as double. Thus, 2 approaches are tested with the same random condition and GPA. The query statement with criteria is shown in Fig. 6 for HQL query and Fig. 7 for Criteria API. The example is

The operator means logical operators which are greater than, less than and equal and the randGPA means the gpa's value for comparing. Both are the random number.

When the executions finish, the queried time are recorded to evaluate. The performance will be compared by the average query time different. The average query time different is calculated by the time difference between 2 approaches. The calculation of the time difference is calculated by the equation shown in Fig. 8.

When the percentage is positive which means the HQL’s query time is less than Criteria API’s query time. Then, the HQL method queries the records set faster than Criteria API method and vice versa.

IV. PRELIMINARY RESULTS
The percentages of time different are compared for the performance of each query technique.

A. The Results of Query without Criteria Case
1) The percentage of time difference of small queried records set between 30,000 and 100,000 records in the database

<table>
<thead>
<tr>
<th>Queryed records set</th>
<th>Average queried time of HQL method (ms.)</th>
<th>Average queried time of Criteria API method (ms.)</th>
<th>The percentage of time difference between HQL and Criteria API methods</th>
<th>S.D. of time difference between HQL and Criteria API methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>6.985</td>
<td>6.98</td>
<td>0.5005</td>
<td>19.270</td>
</tr>
<tr>
<td>500</td>
<td>25.53</td>
<td>25.85</td>
<td>0.6404</td>
<td>10.757</td>
</tr>
<tr>
<td>1000</td>
<td>48.79</td>
<td>48.798</td>
<td>-0.3686</td>
<td>9.242</td>
</tr>
</tbody>
</table>

Fig. 9. The percentages of time difference between HQL query and Criteria API with small queried data of 30,000 records in the database.

Fig. 9. illustrates the decreasing trend of the percentage of time difference. When the data are queried between 100-500 records, the HQL query is worked fast. After the queried records increased, the results are similar to zero. So, both approaches return the same performance.
b) The small queried records set of 100,000 records in the database

TABLE II: THE EXECUTION TIME VALUES OF HQL QUERY AND CRITERIA API WITH SMALL QUERIED DATA OF 100,000 RECORDS SET IN THE DATABASE

<table>
<thead>
<tr>
<th>Queried records set</th>
<th>Average queried time of HQL method (ms.)</th>
<th>Average queried time of Criteria API method (ms.)</th>
<th>The percentage of time difference between HQL and Criteria API methods</th>
<th>S.D. of Time difference between HQL and Criteria API methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>6.98</td>
<td>6.997</td>
<td>-0.3631</td>
<td>14.814</td>
</tr>
<tr>
<td>500</td>
<td>24.929</td>
<td>25.231</td>
<td>0.3183</td>
<td>11.313</td>
</tr>
<tr>
<td>1000</td>
<td>49.186</td>
<td>48.837</td>
<td>-0.397</td>
<td>6.727</td>
</tr>
</tbody>
</table>

Fig. 10. The percentages of time difference between HQL query and criteria API with small queried data of 100,000 records in the database.

TABLE III: THE EXECUTION TIME VALUES OF HQL QUERY AND CRITERIA API WITH LARGE QUERIED DATA OF 30,000 RECORDS SET IN THE DATABASE

<table>
<thead>
<tr>
<th>Queried records set</th>
<th>Average queried time of HQL method (ms.)</th>
<th>Average queried time of Criteria API method (ms.)</th>
<th>The percentage of time difference between HQL and Criteria API methods</th>
<th>S.D. of Time difference between HQL and Criteria API methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>48.721</td>
<td>48.798</td>
<td>-0.369</td>
<td>9.242</td>
</tr>
<tr>
<td>5000</td>
<td>222.276</td>
<td>222.901</td>
<td>0.259</td>
<td>2.215</td>
</tr>
</tbody>
</table>

The percentage difference quite stable when queried the large data. Both figures illustrated that the range around 1,000-3,000 queried records are proper with HQL query method.

b) The large queried records set of 100,000 records in the database

TABLE IV: THE EXECUTION TIME VALUES OF HQL QUERY AND CRITERIA API WITH LARGE QUERIED DATA OF 100,000 RECORDS SET IN THE DATABASE

<table>
<thead>
<tr>
<th>Queried records set</th>
<th>Average queried time of HQL method (ms.)</th>
<th>Average queried time of Criteria API method (ms.)</th>
<th>The percentage of time difference between HQL and Criteria API methods</th>
<th>S.D. of Time difference between HQL and Criteria API methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>49.186</td>
<td>48.837</td>
<td>-0.397</td>
<td>6.727</td>
</tr>
<tr>
<td>5000</td>
<td>222.911</td>
<td>223.187</td>
<td>0.091</td>
<td>2.765</td>
</tr>
<tr>
<td>10000</td>
<td>439.536</td>
<td>441.070</td>
<td>0.333</td>
<td>1.29</td>
</tr>
<tr>
<td>20000</td>
<td>856.021</td>
<td>857.151</td>
<td>0.13</td>
<td>0.705</td>
</tr>
<tr>
<td>30000</td>
<td>1270.476</td>
<td>1271.469</td>
<td>0.076</td>
<td>0.65</td>
</tr>
</tbody>
</table>

The Fig. 9, Table I and Fig. 10, Table II of without Criteria method with small queried records set demonstrated that an appropriate approach depends on the difference environment and the amount of queried data. The HQL query method is suitable with 30,000 records in the database when queried data are not more than 500 records set because it return the better query time around 0.5 percent. After that, the percentage was almost near zero. So, the developers can use both methods. The developers can use both methods. On the other hand, the Criteria API method is suitable for 100,000 records in the database. Only the range between 500-700 records is worked well with HQL query method.

2) The percentage of time difference of large queried records set between 30,000 and 100,000 records in the database

a) The large queried records set of 30,000 records in the database

The Fig. 11 showed that Criteria API method can retrieved the 1,000 and 2,000 records set well. After the queried records increased, the HQL query method is appropriate.

b) The large queried records set of 100,000 records in the database

The Fig. 12 showed that both query methods can be used. 1,000-3,000 queried records are suitable with Criteria method. Then, 4,000-30,000 queried records are proper with HQL query method.

The Fig.11. is similar to the Fig. 12. The values are shown in Table III and Table IV respectively. The values of percentage difference quite stable when queried the large data. Both figures illustrated that the range around 1,000 -3,000 queried records work properly with Criteria API methods. When the queried records raised, the percentage of time difference values are increased. Therefore, the appropriate method is HQL query.

The Fig. 10 shows that both approaches are optimal in the environment of 100,000 records in the database because the values are positive and negative in queried data ranges. So, 100-200 and 800-1,000 records worked with Criteria API faster than HQL query, and the others queried fast with HQL query. Moreover, 300-400 can use both methods.
B. The Results of Query with Criteria Case

1) The percentage of time difference of small queried records set between 30,000 and 100,000 records in the database

a) The small queried records set of 30,000 records in the database

Then, the queried records raised, the developers can use both methods at 800 - 900 records set since the values are near zero. However, 700 and 1,000 records set can use HQL query method. In Fig. 13 and TABLE V described the suitable query mechanism. Both database environments can be used both query mechanisms. However, some ranges of queried data are used the proper method. For Fig.13. is only at point of 400 records that worked with HQL and point of 800 records that worked with Criteria API. Moreover, the Criteria API method is optimal only between 100 - 600 queried records with 100,000 records in the database.

b) The small queried records set of 100,000 records in the database

When the commands with WHERE condition executed, the percentage of time difference values almost reached zero as show in Fig. 13. The result pretended to use both query mechanism. Accept 400 queried records, the HQL query worked properly. Moreover, 700-800 queried records are proper with Criteria API method.

2) The percentage of time difference of large queried records set between 30,000 and 100,000 records in the database

a) The large queried records set of 30,000 records in the database

When the commands with WHERE condition executed, the percentage of time difference values almost reached zero as show in Fig. 13. The result pretended to use both query mechanism. Accept 400 queried records, the HQL query worked properly. Moreover, 700-800 queried records are proper with Criteria API method.

b) The large queried records set of 100,000 records in the database

When the environment changed to 100,000 records set in the database, most queried records worked with Criteria API
approach executed very well except 1,000 queried records in Fig. 16. It showed the small value of percentage time difference closed to zero. So, the value was not difference significantly, and both methods can be used.

In the experiment with WHERE condition of large queried records set displayed the similar result as shown in Fig. 15, Table VII and Fig. 16, Table VIII. The Criteria API method can be used properly for both environments because the values were minus values and the percentage of time difference values were distinct. The result of this experiment is concluded in Table IX.

<table>
<thead>
<tr>
<th>Query Mechanisms</th>
<th>Command without WHERE condition</th>
<th>Command with WHERE condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Small queried records set (100 – 1,000 records)</td>
<td>Large queried records set (1,000 – 30,000 records)</td>
</tr>
<tr>
<td></td>
<td>30,000 records in the database</td>
<td>100,000 records in the database</td>
</tr>
<tr>
<td>HQL Query</td>
<td>x – only 500-700 queried records set</td>
<td>x – only 1,000-3,000 queried records set</td>
</tr>
<tr>
<td>Criteria API</td>
<td>x – only 1,000-3,000 queried records set</td>
<td>x – only 1,000-4,000 queried records set</td>
</tr>
</tbody>
</table>

x means the optimal mechanism with each environment of queried data and database.

xx means the percentage of time difference value is significant. Thus, the results show the appropriate query approach obviously.

V. CONCLUSION

In the study, the testing of query performance is proposed with Hibernate Framework by HQL query and Criteria API methods. The experiment is tested to show which query mechanism can retrieve data fastest. In this section is described the conclusion of the experiment of difference query mechanisms. The results illustrate the performance of querying the data in difference query mechanisms, number of queried records set and the difference of commands between difference environments. We measured by comparing the percentage of time different. Firstly, the difference structure of commands can represent the optimal method. If the commands without WHERE condition executed, the HQL query worked properly. Since the characteristics of commands without WHERE condition will retrieve all data from the database, the structure of HQL is similar to SQL which get the data easily. On the other side, the Criteria API method can operated commands with WHERE condition. The function of this approach represents each data entity as an object. When the commands with WHERE condition executed, the instantiation of objects can be managed. Therefore, the Criteria API method can deal with many conditions occur. When the number of data is 30,000 records in the database, the results show that the developers can use both query methods in the difference queried ranges. The values of percentage difference are not distinct because the values are almost zero. However, a suitable approach is HQL query since the commands are similar to SQL statement. The commands are executed very well under the small amount of data. On the contrary, the number of data in the database is increased to 100,000 records, the execution results are stable. Most of them shown negative values which mean the Criteria API method is proper. This method uses the instance of objects to manipulate with related conditions. Therefore, the query mechanism retrieves data by using Object Oriented Programming technique. The developers are familiar and do not waste time to learn commands’ structure. The experiment is executed faster than HQL query. This study summarizes the optimal query mechanisms which show in the appendix. The table describes the difference environment works with difference query approach. The results are clear and understandable.

In the future work, the application will be developed to test about the data complexity and created a new transformation algorithm to transform between HQL commands and Criteria API commands. The algorithm can operate automatically and convert commands in vice versa. The developers can select a query mechanism for their database application. When the database application is developed, the developers are interested a usual query method with their comprehension. Sometimes they want to try a better query mechanism, but it wastes time to study in both the commands.

REFERENCES


Tisinee Surapunt was born in Chiang Mai, Thailand, on April 30th, 1988. The background education is Bachelor of Science, software engineering major (International Program) of College of Arts Media and Technology, Chiang Mai University, Chiang Mai, Thailand in 2009.

Presently, she is a graduate student in software engineering major of College of Arts Media and Technology, Chiang Mai University, Chiang Mai, Thailand. Moreover, she works as a teaching assistance in software engineering program of bachelor degree, College of Arts Media and Technology, Chiang Mai University, Thailand. Her research interests are in testing field, also the advertent performance testing.

Ms. Surapunt was awarded an Erasmus Mundus Scholarship to study at Corvinus University of Budapest, Budapest, Hungary in October 2009.