ISSN: 0974-6471, Vol. 11, No. (2) 2018, Pg. 115-125

Oriental Journal of Computer Science and Technology

Journal Website: www.computerscijournal.org

Evaluating Performance of Hibernate ORM based Applications using HQL Query Optimization

SALAHUDDIN SADDAR, JUNAID BALOCH, MEMOONA SAMI*, NASRULLAH PIRZADA, VIJDAN KHALIQUE and ARSALAN AFTAB MEMON

Mehran University of Engineering & Technology Jamshoro, Hyderabad, Pakistan.

Abstract

With the continuous advancement in technology, web technologies have reached to a new height. Enterprise applications are meant to be the basic need of today's world which aims to produce results that are highly reliable, portable and adaptable. With these enormous features, they needed the storage mechanism which could handle and store the data effectively. The storage system thus required was a database management system but again technical knowledge was required to make things work appropriately. However, this approach was the traditional one which requires data to be stored in tabular format whereas Object oriented architecture has taken the programming language to a whole new technical phase for which the traditional RDBMS will not efficiently accomplish the expected job. To fulfill this gap in the literature, Object Relational Mapping is emerged as a solution to provide which provide comparative technical features effortlessly. These characteristics simplify and make the mapping of objects in object-oriented programming languages more flexible, efficient and easy to use. Therefore, we propose in this paper that Object relational Modeling (ORM) relates each object of object oriented languages to corresponding rows in the table.

Article History

Received: 24 April 2018
Accepted: 29 May 2018

Keywords

Hibernate,
ORM,
HQL,
Mapping,
RDMS,
OOP.

Introduction

On the temporary basis, filing and spreadsheets is used for storing massive data. However, commonly relational databases are used for the application and its availability. Relational databases require data to be stored in form of tables which is not sufficient. It does not work out with this fast emerging technical world and its needs. Since, now the time has gone where procedural languages were used. Today is an era of Object oriented programming which considers each and everything as an object and serves better for providing enterprise solutions.

The selection for Java language is justified since it is an open source language and also provides number of facilities such as: an API for querying and manipulating databases commonly termed as Java Database Connectivity (JDBC) API. Along with that,

CONTACT Memoona Sami memoona.sami@faculty.muet.edu.pk Mehran University of Engineering & Technology Jamshoro, Hyderabad, Pakistan.

To link to this article: http://dx.doi.org/10.13005/ojcst11.02.07

it gives the important transformation of information between the two sorts (JAVA and RDBMS)[1].

Also, there are number of ORM instruments to play out the activity by coding.

Literature Review

In modern development, Object Relational Mapping such as Hibernate has proved to be a good choice. Due to its portable and easy to use nature, it has been popular[2]. Different languages are pooled in a database application for data processing. They proposed a framework to optimized database applications which proves to be a significant solution in real world applications[15]. Various factors such as database access and efficiency influence Information management systems. A real case is enlightened to define hibernate ORM mechanisms[16]. ORM tools are used to map objects to and from relational form. This paper highlights the evaluation performance of open source object persistence[17].

Our proposed study examines the ORM tool by performing distinctive tests, in various situations, this examination is centered on a wide assortment of hibernate consideration. Execution time is the basic measurement tool for it.

- It considers numerous situations especially related about one-to-many relation in hibernate.
- Correlates object database and ORM tools.
- Examines an ORM tool hibernates by performing distinctive investigations, in various situations.
- Has pleasantly centered on execution improvements.
- Drives attention on number of inquiries executed however following are also significant performance concerns.

ORM v/s Versant Object DB

Versant has more than 50000 users from different fields, like government, medical and telecommunication.

However, hibernate is a standard tool, used for ORM queries. It can be used with a relational database as well as with Oracle Enterprise edition.

Table 1: Comparison between Hibernate and Versant

Functionality	Hibernate	Versant
Schema Generation	Yes	Yes
Java Interfaces	Yes	Yes
Bi-directional relationships	Yes	Yes
Session Control	Yes	Yes
Locking Mechanisms	Yes	Yes
Optimization	Yes	Yes
Database Access Control	Yes	Yes
XML Support	Yes	Yes In some cases
Distribution	Yes	Yes

A basic investigation was carried out to judge the performance of object database with ORM tools by making use of tools like Versant Object Database Management System (ODBMS) and Hibernate.

The comparison between features is shown in table 1.

The main concept behind the speed of hibernate is that the query is carried out on indexed columns of objects which ultimately makes hibernate much faster than versant so it has won this race in terms of performance.Both tools provide java interfaces, sessional control mechanism. Hibernate offers XML support in all cases but versant supports for some cases.

Hibernate v/s JPOX

This study examines the functioning of Hibernate and OpenJPA and JPOX and Speedo.

Diverse sort of tests were carried out that were greatly based upon and data retrieval. The following queries were utilized for execution tests.

- To load one object twice within a transaction.
- To load a single object two time among these transactions.
- Lazy loading an object data.
- Eager loading an object data.

- Loading and saving similar and complex objects.

Processing time gives the immediate execution consequences and shows performance of ORM or persistence device for a particular test[5].

The consequences of the lazy loading were better in the case of the Hibernate and OpenJPA than the JPOX and Speedo.

If the second query is concerned, ORM apparatus performed exceptional. The high hand of caching systems was proved to take place in this regard[3].

So finally, it is concluded that hibernate played out the best general in majority of queries however, JDO marked its way in case of eager loading and complex objects.

Hibernate v/s iBatis

Discussions were carried out in which examination of iBatis and Hibernate was embraced. With a specific end goal to play out the correlation, different tests were performed using distinctive exercises on a small banking system. The tests are dependent on a Java program, which utilizes both iBatis and Hibernate. It performs essential Structured Query Languages (SQL) operations as execution test. Theseexperiments were executed for numerous clients. Java threads were carried out for multi-client tests. To play out the tests the following input sources can be utilized[4].

- Selectcontinues used tool.
- Records which are to be updated at one time
- Select the need of performing the entire or one of the operations.
- Which threads (users) used to execute the multiuser testing.
- Which iterations mean the number of the time a particular set has been accessed?

The outcomes demonstrate that Hibernate utilizes much time to embed information because of mapping. Likewise, it was revealed that Hibernate consumes extra time to run out the query for the 1st time then the 2nd time. This is essentially because of the store cache implementation in Hibernate. Hence it is proved; Hibernate performs superior to iBatis in hot queries, while performs slower for cold queries.

Hibernating

Hibernate ORM (Object Relational Model) is a mediator between Java based objects and relational database which shapes the entities into object oriented domain. Hibernation enables malleable access to relational databases with high level object based functions.

The Figure 1 shows the layered architecture of Hibernate, which is used to help the users without knowing APIs. Hibernate uses database and configuration data to deliver services to the applications.

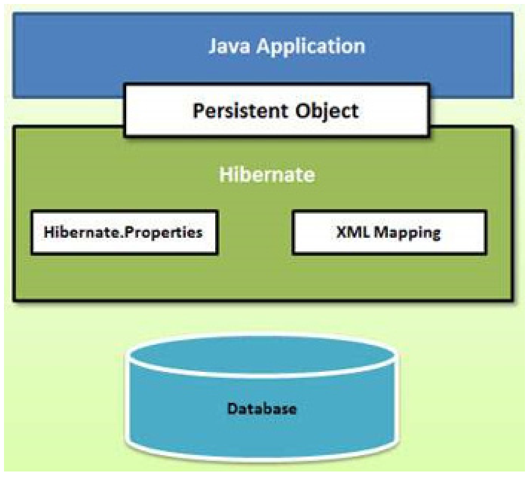


Fig.1: Structure of hibernating

HQL (Hibernate Query Language) and SQL (Structured Query Language) are almost similar. HQL, however, deals with objects and SQL works on tables. To carry out the interactions with relational databases, Hibernate transforms HQL queries into SQL.Not only this, Hibernate is itself capable of executive direct SQL queries. HQL is much preferred in order to avoid database portability issues which can arise in near future[7].

HQL Clauses
Keywords such as SELECT, FROM and WHERE and so forth are utilized as Keyword for writing clauses to perform HQL operations.

FROM Clause
Persistent objects are loaded using the From Clause into the memory. It requires Package and Class names.

AS Clause
In the HQL queries, AS clause is utilized to allocate Aliases to the classes. This clause proves to be useful to manage long queries. The Aliases can be relegated to the classes specifically.

SELECT Clause
The select condition empowers the designer to recover specific properties of the items instead of the entire question itself. The Select condition empowers the engineers to apply more control over the choice contrasted with the FROM statement[13].

WHERE Clause
Where provision empowers the engineer to characterize a condition in view of which the properties ought to be recovered, consequently empowering to limit the outcome.

ORDER BY Clause
This statement is used to select the outcomes which arerecovered by the inquiry. The requesting/arranging of the outcomes relies upon the boundary gave by the engineer which can either be DESC for descending order or ASC for ascending order.

UPDATE Clause
Engineers may utilize the Update Clause to refresh the properties of an object, the e. With the Hibernate 3, mass updates can be performed. With a specific end goal to execute the update, the execute Update() method is utilized[6].

DELETE Clause
Delete clause is utilized by the designer to erase at least one object. The erasure of objects works contrastingly by comparing Hibernate 2 and 3. The execute Update() method is utilized for deleting purpose,.

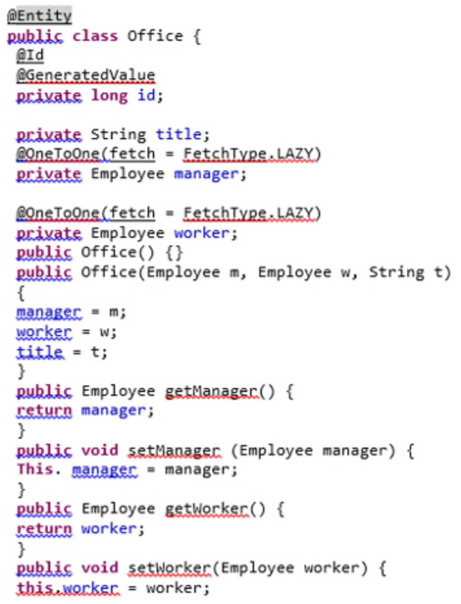
INSERT Clause
This clause is utilized to embed new records in HQL.

Object Relational Mapping (ORM)
Object Relational Mapping is a strategy that is utilized to bring similarity between the distinctive sorts of information. If an object is stored into data store and converted back to object without changing its properties, it is said to be persistent[8,14].

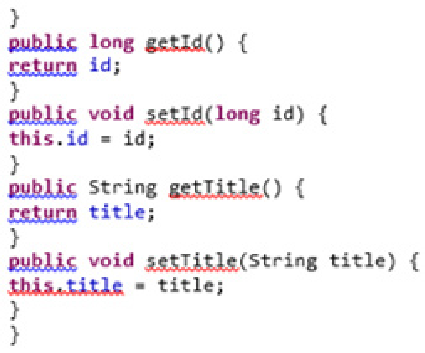
One of the Code bits for this is:-

A case of Office and Employee class is taken as an example.

The above mentioned code in figure 2 (b)is used to create a JAVA Office dialect class.The @Entity comment performs as an entity class. To generate

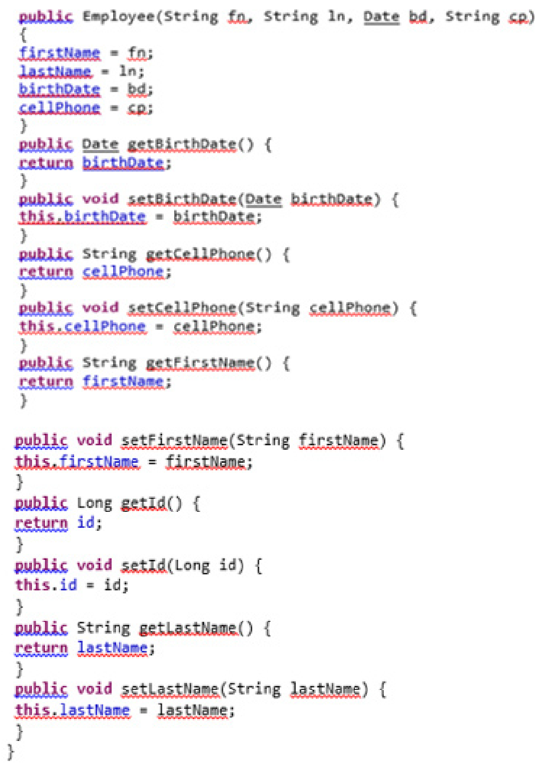


```
@Entity
public class Office {
 @Id
 @GeneratedValue
 private long id;

 private String title;
 @OneToOne(fetch = FetchType.LAZY)
 private Employee manager;

 @OneToOne(fetch = FetchType.LAZY)
 private Employee worker;
 public Office() {}
 public Office(Employee m, Employee w, String t)
 {
 manager = m;
 worker = w;
 title = t;
 }
 public Employee getManager() {
 return manager;
 }
 public void setManager (Employee manager) {
 This. manager = manager;
 }
 public Employee getWorker() {
 return worker;
 }
 public void setWorker(Employee worker) {
 this.worker = worker;
```



```
}
public long getId() {
return id;
}
public void setId(long id) {
this.id = id;
}
public String getTitle() {
return title;
}
public void setTitle(String title) {
this.title = title;
}
}
```

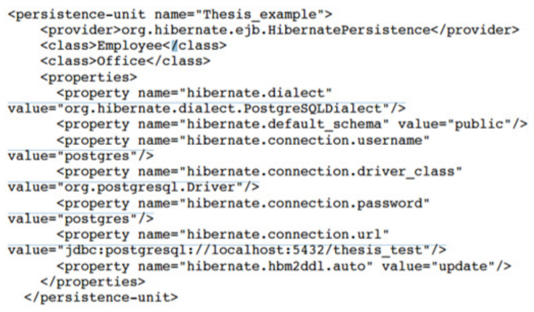
Fig. 2 (a) Office and Employee class code



```
public Employee(String fn, String ln, Date bd, String cp)
{
firstName = fn;
lastName = ln;
birthDate = bd;
cellPhone = cp;
}
public Date getBirthDate() {
return birthDate;
}
public void setBirthDate(Date birthDate) {
this.birthDate = birthDate;
}
public String getCellPhone() {
return cellPhone;
}
public void setCellPhone(String cellPhone) {
this.cellPhone = cellPhone;
}
public String getFirstName() {
return firstName;
}

public void setFirstName(String firstName) {
this.firstName = firstName;
}
public Long getId() {
return id;
}
public void setId(Long id) {
this.id = id;
}
public String getLastName() {
return lastName;
}
public void setLastName(String lastName) {
this.lastName = lastName;
}
}
```

Fig: 2(b) JAVA Office dialect class

keys automatically the @id and @Generated Value is utilized they are also used to represent the primary key. @OneToOne comment gives affiliation the Employee class instances i.e. worker and manager. Additionally the Employee class gives the data to create the employee instance. The code bit in Figure 3 displays how to make object relational mapping between a relational database and a programming dialect like JAVA.



```
<persistence-unit name="Thesis_example">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <class>Employee</class>
    <class>Office</class>
    <properties>
      <property name="hibernate.dialect"
value="org.hibernate.dialect.PostgreSQLDialect"/>
      <property name="hibernate.default_schema" value="public"/>
      <property name="hibernate.connection.username"
value="postgres"/>
      <property name="hibernate.connection.driver_class"
value="org.postgresql.Driver"/>
      <property name="hibernate.connection.password"
value="postgres"/>
      <property name="hibernate.connection.url"
value="jdbc:postgresql://localhost:5432/thesis_test"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
    </properties>
  </persistence-unit>
```

Fig. 3: Object relational mapping between a relational databaseand a programming dialect like JAVA

Problems of Data Navigation

Java and relational databases both deal differently in getting to information and navigating or exploring from one object to next[11]. The associations can be either unidirectional or bidirectional. Java accesses data by making use of getter methods, e.g., getName(). However, getting to a particular data in a relational database will require SQL queries which can be complicated much if it requires joins. The performance can only be enhanced if minimum requests are made to database[9].

Inheritance Mapping

Mapping of inheritance particularly defined at the object level is the real essence of inheritance mapping. To map a database scheme, three various inheritance techniques are listed out:

Inheritance Mapping to Single Table: Nulls' Style
A class with its entire subclasses is mapped to a single database table which can possible nullify few attributes in that table. That's the reason this type of mapping is referred to as Nulls' style.

Inheritance Mapping to Multiple Style: ER Style

This technique plots a segment to no of tables in the database of the inheritance hierarchy.

Association Mapping

Association is the basic relation or affiliation between the two classes or the collection between the classes.

Three kinds of association mappings are defined below:

MANY-TO-ONE ASSOCIATION

An association L on class M and N is many-to-one if for any object M, is related with at most one object in N.

ONE-TO-ONE ASSOCIATION

L will be a balanced association, if any object in N is related with at most one object in M[9].

MANY-TO-MANY ASSOCIATION

If none of above mentioned condition matches, it falls in the category of many-to-many association[12].

Desinging Experimental Senarios

The few Test cases scenarios are illustrated below: Table 2,3,4 shows the test cases.

Table 2: HQL Query Optimization

Test Case #: 001 **Test Case Name: Query Loading**
System: Hibernate **Subsystem: HQL Query Optimization**
Designed by: MemoonaJunaid **Design Date:12/09/2017**
Executed by: SalahuddinSaddar **Execution Date:12/09/2017**
Short Description: Testing SELECT and JOINS Queries
Pre-Conditions:

- **Setting environments for querying in IDE'S**
- **Server must be running**
- **Designed database in MySQL**

Step	Action	Expected System Response	Actual Result	PASS/FAIL	Comment
1	Loaded Large SELECT Queries in comparison to other tools	Least time to fetch all records	Took 152 ms to fetch all the records	PASS	Verified
2	Loaded Small SELECT Queries in comparison to other tools	Least time to fetch all records	Took less time from some of the other tools	PASS	Verified
3	Loaded joins	Small execution time	Takes long time	PASS	Verified

Test Case #: 002 **Test Case Name: Criteria API**
System: Hibernate **Subsystem: HQL Query Optimization**
Designed by: MemoonaJunaid **Design Date:12/09/2017**
Executed by: SalahuddinSaddar **Execution Date:12/09/2017**
Short Description: Criteria API instead of HQL
Pre-Conditions: Testing Criteria API

- **Setting environments for querying in IDE'S**
- **Designed Databases in MySQL**
- **Server must be running**

Step	Action	Expected System Response	Actual Result	PASS/FAIL	Comment
1	HQL Query	Retrieved the result set in lists containing all the data	Retrieved the result set in lists containing all the data	PASS	Verified
2	Query using Criteria API	Retrieved the result set as a complete model of bean class	Retrieved the result set as a complete model of bean class	PASS	Verified

Table 3: Fetch type LAZY

Test Case #: 003 **Test Case Name: Fetch type LAZY**
System: Hibernate **Subsystem: HQL Query Optimization**
Designed by: MemoonaJunaid **Design Date:12/09/2017**
Executed by: SalahuddinSaddar **Execution Date:12/09/2017**
Short Description: Fetching data using LAZY annotation
Pre-Conditions:

- **Setting environments for querying in IDE'S**
- **Server must be running**
- **Designed database in MySQL**

Step	Action	Expected System Response	Actual Result	PASS/FAIL	Comment
1	Fetching using type EAGER	Will result all the objects initially	Retrieved all the objects, resulting in high throughput	PASS	Verified
2	Fetching using type LAZY	Will result the objects and data	Retrieved desired objects resulting low throughput requested	PASS	Verified

Experiments and Results
Experiment 1
Objective

Fetching Parameter Eager Using Hql Query Option

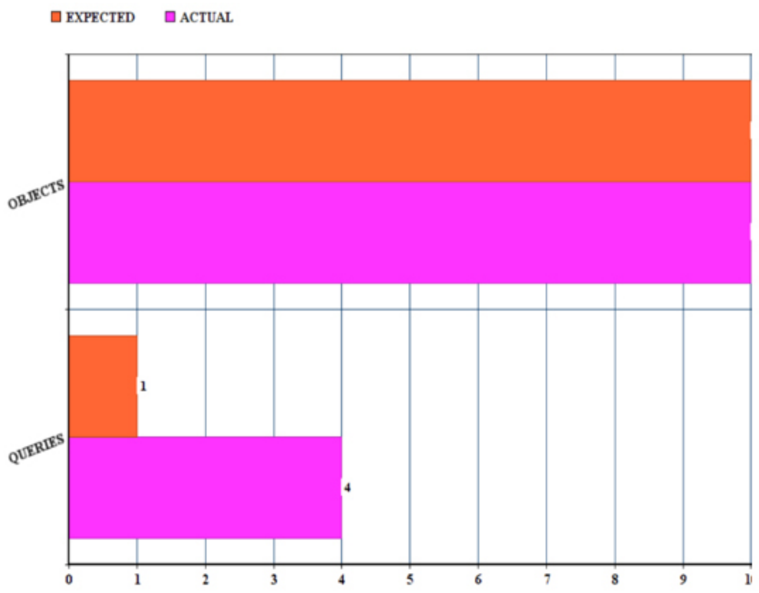


Fig. 4: Fetch Parameter Eager With Hql Query Option

Result

As the table shows, the normal and the real records are same, using fetch type Eager, Hibernate is expressly characterized to collect entire records along with the accumulations of entire objects.

The time efficiency of these two is portrayed as:

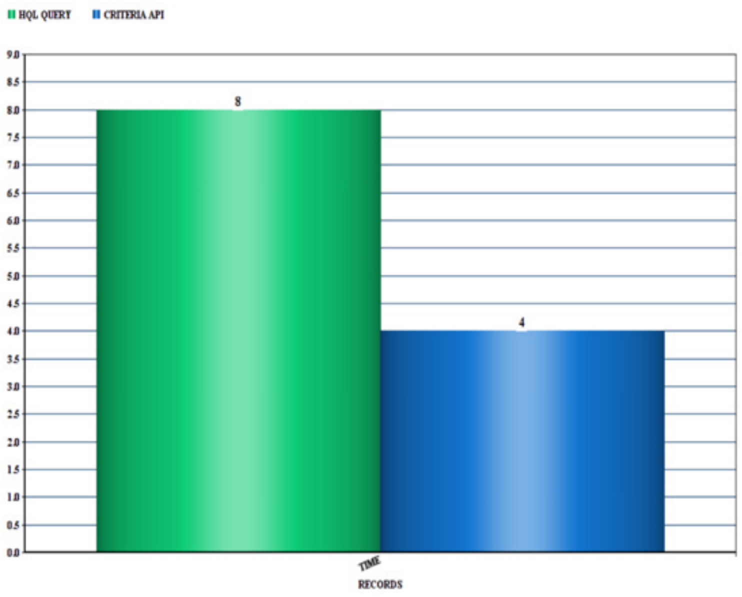


Fig. 5: Time Efficiency of HQL Query and Criteria API

Experiment 2

Objective

Associated Collections Are Loaded Via Useless Database Hit Even in The Absence of Associated Object Exists at All

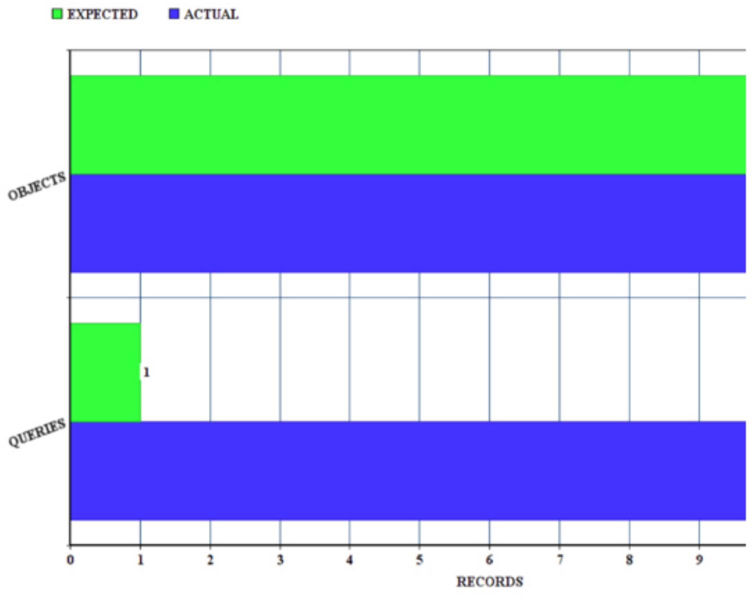


Fig. 6: Useless Database Hit To Load Associated Collection

Result
As a result of Eager search, a considerable measure of queries was characterized by HQL, as the majority of the objects are retrieved in a single run in view of the preparing of extensive number of queries.

Experiment 3
Objective
Associated collections are not fetched by HQL query joins. Extrahits are obligatory to load theassociation.

Result
As the outcome table displays, the normal queries count are ZERO. But, but it is not real scenario on account of the Fetch type utilized is Eager, as specified above. Which bring about stacking of total objects, in addition to the index and the different gathering of objects?

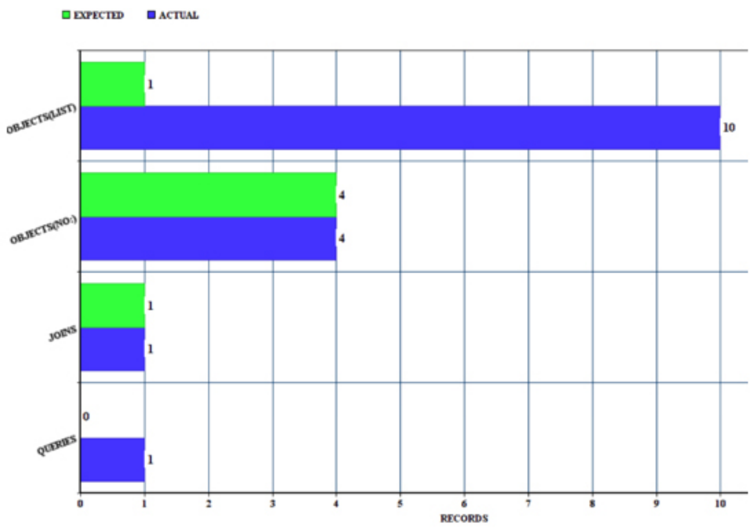


Fig. 7(A): Hql Query Joins Fetching

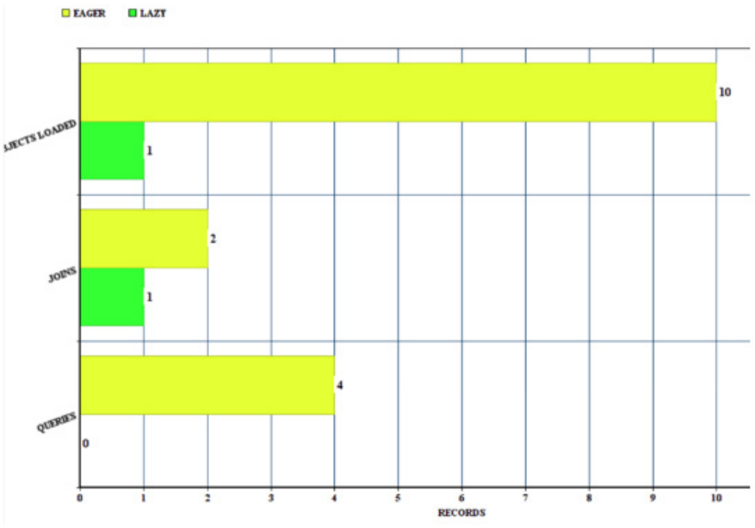


Fig. 7 (B) Hql Query Joins Fetchng The Associated Collections

Benefits

- No any master required for database administration.
- Avoids superfluous access to databases.
- Avoids composing of long queries and joins.
- Suitable for an expansive enterprise applications.
- Requires no progressions to be made, if databases are changed or the other way around.

Conclusion

There are different techniques for managing transactions implemented by systems that can affect system performances. This research basically focuses on how each system performs when recovering data, that's all important to focus on factors such as cache management, query language, and ORM functions that have these tools. Similarly, we used different types of searches to check the operation of different systems run. These queries are selected according to different criteria and navigation paths. These questions provide a way to check the different functions of systems, such as their search languages inheritance strategies and their caching mechanisms. These functions can help us decide which The system works better and what needs to be done to improve performance. It is very much clear from the results that utilizing Criteria API rather than conventional HQL Query will furnish result sets with wanted throughput that will upgrade the execution of querying. Moreover, recovering records utilizing LAZY fetch type will give low overhead to applications. A good ORM will automatically generate all the SQL needed to store the objects.

Acknowledgement

We would like to thanks co-authors for the discussion and contributions in provided research.

References

1. Chen, Tse-Hsun, *et al.,* "An empirical study on the practice of maintaining object-relational mapping code in java systems." *Proceedings of the 13th International Conference on Mining Software Repositories.* ACM, 2016.
2. Decan, Alexandre, Mathieu Goeminne, and Tom Mens. "On the interaction of relational database access technologies in open source java projects." arXiv preprint arXiv:1701.00416(2017).
3. Dhingra, Neha, EmadAbdelmoghith, and Hussein T. Mouftah. "Performance Evaluation of JPA Based ORM Techniques." In Proceedings of 2nd International Conference on Computer Science Networks and Information Technology, Held on 27th - 28th Aug 2016.
4. Babu, Chitra, and G. Gunasingh. "DESH: Database evaluation system with hibernate ORM framework." Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on. IEEE, 2016.
5. Fraczek, Konrad, and MalgorzataPlechawska-Wojcik. "Comparative Analysis of Relational and Non-relational Databases in the Context of Performance in Web Applications." International Conference: Beyond Databases, Architectures and Structures. Springer, Cham, 2017.
6. Wan, Jing-Jing, Zhen Qin, and Xia Liu. "ORM Elevation in Response to Cognitive Impairment Is an Accompanying Phenomenon." *CNS neuroscience & therapeutics* 22.8 (2016): 723-724.
7. Bernstein, Philip A., *et al.,* Incremental mapping compilation in an object-to-relational mapping system (extended version). Technical Report MSR-TR-2013-45, Microsoft Research, 2013.
8. Chen, Tse-Hsun, *et al.,* "CacheOptimizer: Helping developers configure caching frameworks for Hibernate-based database-centric web applications." *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering.* ACM, 2016.
9. Procaccianti, Giuseppe, Patricia Lago, and WouterDiesveld. "Energy efficiency of orm approaches: an empirical evaluation." Proceedings of the 10th ACM/IEEE *International Symposium on Empirical Software Engineering and Measurement.*

ACM, 2016.

10. Chen, Tse-Hsun, *et al.,* "Cache Optimizer: Helping developers configure caching frameworks for Hibernate-based database-centric web applications." *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering.* ACM, 2016.
11. Chen, Tse-Hsun, *et al.,* "Detecting problems in the database access code of large scale systems-an industrial experience report." Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on. IEEE, 2016.
12. Chen, Tse-Hsun, *et al.,* "An empirical study on the practice of maintaining object-relational mapping code in java systems." *Proceedings of the 13th International Conference on Mining Software Repositories.* ACM, 2016.
13. Grechanik, Mark, *et al.,* "Enhancing rules for cloud resource provisioning *via* learned software performance models." *Proceedings* of the 7th ACM/SPEC on International Conference on Performance Engineering. ACM, 2016.
14. Ackermann, Hilmar, *et al.,* "A backend extension mechanism for PQL/Java with free run-time optimisation." International Conference on Compiler Construction. Springer, Berlin, Heidelberg, 2015.
15. Emani, K. Venkatesh, and S. Sudarshan. "Cobra: A Framework for Cost Based Rewriting of Database Applications." arXiv preprint arXiv:1801.04891 (2018).
16. Zhengju, Chen. "Hibernate-Based Database Access Optimization [J]." *Computer Applications and Software* 7 (2012): 045.
17. Van Zyl, Pieter, Derrick G. Kourie, and Andrew Boake. "Comparing the performance of object databases and ORM tools." *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries.* South African Institute for Computer Scientists and Information Technologists, 2006.