Hibernate ORM Query Simplication Using Hibernate Criteria Extension (HCE)

Kisman
Master of Information Technology
Bina Nusantara University
Jl. Kebon Jeruk Raya No. 27, Jakarta Barat, DKI
Jakarta, Indonesia 11530
kismanhong@gmail.com

Sani M. Isa
Master in Computer Science
Bina Nusantara University
Jl. Kebon Jeruk Raya No. 27, Jakarta Barat, DKI
Jakarta, Indonesia 11530
sani.m.isa@binus.ac.id

***Abstract*— Software development time is a critical issue in software development process, hibernate has been widely used to increase development speed. It is used in database manipulation layer. This research develops a library to simplify hibernate criteria. The library that is called as Hibernate Criteria Extension (HCE) provides API functions to simplify code and easily to be used. Query associations can be defined by using dot. The library will automatically detect the join association(s) based on mapping in entity class. It can also be used in restriction and order. HCE is a hibernate wrapper library. The configuration is based on hibernate configuration. This library can be easily used by the developers whom familiar with hibernate usage.**

Keywords*— *database; java; hibernate; hce; query; simplify

I. INTRODUCTION

Nowadays, most of software application use database for storing information/data. Database and database systems are essential component of life in modern society: most of us encounter several activities every day that involve some interaction with a database [1]. Applications store and access the information in a textual or numeric type. This is called as **traditional database application** [1]. The other usages of database are: **Geographic Information Systems (GIS)** that store and analyze maps, **Online Analytical Processing (OLAP)** that is used to extract and analyze useful information from very large databases to support decision making.

Database is a collection of related data. The data is recorded and has an implicit meaning. The collection of data usually called as database. A database, is in essence a collection of information that will exist in a period of time, information that is managed for a RDBMS [2].

This paper is intensively discussing about communication between application and database. A software engineer needs to get and put data to the database system, this process step is called querying. Every action that interacts with database's data will be interfaced by a query. The software engineer will make the query specified to database used. Each database vendor has their Structured Query Language (SQL). As the development of software technology and most of programming languages are object oriented, some engineer or software institutions try to simplify the query process. They try to bind object in application to database. This approach is called as Object Relational Mapping (ORM). ORM is a translation mechanism from object to relational data, vice versa. ORM has "dialect" which is defined for generating SQL query to a specified SQL vendor. By using dialect, application will not be restricted by SQL specific database vendor. Application developers can choose any database to be used with minimum changes. This is one of the application and database communication problems for application that is designed can be used with any database vendor.

To simulate on how ORM work and simplify ORM usage, this research use Java and Hibernate ORM framework. This research develops a tool/library that is developed as a hibernate wrapper library called as **Hibernate Criteria Extension (HCE)**. This research develops HCE as wrapper library based on hibernate. The author will develop code to be a java library (JAR). The library code is developed using hibernate library and its dependencies. HCE contains of Java API functions. The Java software engineer who are familiar with hibernate can easily understand the purpose of HCE. We can simplify query that is already simplified by hibernate. The reasons on why hibernate is chosen as core of HCE are: Hibernate is a popular Java ORM framework, hibernate has criteria API that the others ORM don't. The other Java ORM frameworks are: DataNucleus, Avaje Ebean, MyBatis, Apache Cayene.

Hibernate Criteria Extension (HCE) is a Java library on top of hibernate. HCE provides many API functions to simplify hibernate functions, especially the criteria API functions. This library uses many functions of criteria API. HCE can detect relation automatically from the hibernate configuration at Java Object Class/Plain

978-1-5090-2100-0/16/$31.00 ©2016 IEEE

Java Object (POJO). We need only to mention the field (refers to column) and which class (refers to table) to be queried. For relation/join usages, what we need only to state is the field's name by using ".". We can easily construct the where clause statement, too. The result can be returned by its default or based on the passing parameter by the code. These HCE's features let us save software development time, line of codes and reducing software development cost. We can also let user use the application on time. The complete explanation will be discussed at the Hibernate Criteria Extension (HCE) section.

II. Object-Relational Database Management Systems

Most current database systems are based on the relational model, which was proposed in scientific publication by E. F. Codd in 1970, with the intention of providing a basis for data independence. The relational model is based on two concepts, relation and table, which differ in their nature but are highly related [3].

According to the relational model all data is represented as mathematical n-array relations. A relation is defined as a set of n-tuples. A tuple is an unordered set of attribute values and an attribute is an ordered pair of attribute name and type name. The relational model allows the data be operated upon by means of relational operations of restriction, projection, Cartesian product and relational join. The relational model allows you to define data structures and constraints that guarantee the integrity of the data. (Bauer, C. & King, G. 2007) [4]

In mathematic view, lets us assume two sets, D_1 and D_2 (D is domain). The *cartesian product* of D_1 and D_2 is sets of ordered pairs ($e1$, $e2$) where $e1$ is an element of D_1 and e_2 is an element of D_2. For example, given the sets J={1,2,3} and K={c,d}, the cartesian product of J X K is the set the combination pairs which first element is member of J and the second is K. The combination sets will be:

{(1,c), (1,d), (2,c), (2,d), (3,c), (3,d)}

From the example shown above, a relation of sets D_1 and D_2 is a subset of D_1 X D_2. [3]

Figure 1 A catersian product and a relation

A database management system (DBMS) is an aggregate of data, hardware, software, and users that helps an enterprise manage its operational data [5].

The improvement of DBMS known as Relational Database Management System (RDBMS). RDBMS is a powerful database technology to be used for storing and manipulating transactional data. The consistency and data security are kept well though the database is accessed by multiple user simultaneously. The powerful of RDBMS also imposes limitations, such as:

- Data Complexity: data in RDBMS resides in multiple tables, which are linked to each other through shared key values. RDBMS does not force engineer to impose coherent data structure, inexperience engineer may design systems that create unnecessary complexity or limit the future development.
- Broken Keys and Record: Relational databases require shared keys to link information spread across several tables. For example, a customer table may include client demographics, with a unique index number identifying the record within the table. If the data types linking the keys are different, the database cannot link the records without additional rework by the report developer. Likewise, if a table lacks a unique key, the database may return inaccurate results. If the application accessing a database isn't coded to lock records during an edit, users could inadvertently corrupt data, leading to broken records.
- Developer Expertise: As the complexity of a relational database increases, the skill set required by the RDBMS administrator, various users and report developers also increases.
- Hardware Performance: Complex queries require sophisticated processing power. Although most desktop computers can manage the databases of the size and complexity often encountered in a small business setting, a database with external data sources or very complex data structures may require more powerful servers to return results within an acceptable response time.

Object relational database management systems (ORDBMS) is an extension of relational database management systems, which added object oriented features or direct representation of application objects in relational databases. [6] ORDBMS is a good choice for complex data processing when traditional SQL DBMSs is not capable to handle.

In Java application, the connection to the database function is known as Java Database Connectivity (JDBC). JDBC is needed as a database can interpret only

SQL statements and not Java language statements. JDBC is used to translate Java statement into SQL statements.

III. Object Relational Mapping

Object-oriented programming (OOP) is today the dominant paradigm in mainstream software development, although it had a tentative beginning [7]. Object relational mapping is a technology that is employed to bridge the impedance mismatch between object-oriented programs and relational database. It tries to eliminate the duplication of data and maintenance cost and susceptibility to error(s) associated with it [6]. The impedance mismatch problem is a well-known problem in persistence objects in relational databases between both the object model and the relational model and between the object programming language and the relational query language [6]. Impedance mismatch is a consistency problem between object and database schema design. The object declaration can be more complex. For example, one table and its relation is represented by more than one object. The application will be more complex and difficult to be maintained. A mismatch is addressed using one or more mapping strategies, typically embodied in a pattern. A strategy is concerned with correspondence between the schema of a relational database and an object-oriented program. Such strategies are employed in mapping tools such as Hibernate and TopLink, and reinforce the received wisdom that the problem of object-relational impedance mismatch has been solved. [8]

An object-relational application combines artefacts from both object and relational paradigms. Essentially an object-relational application is one in which a program written using an object-oriented language uses a relational database for storage and retrieval. A programmer must address object-relational impedance mismatch ("impedance mismatch") problems during the production of an object-relational application [9]. For a better understanding of ORM and why ORM is a solution in many software developments, we will try to discuss the JDBC disadvantage and Java ORM popular tool, hibernate.

3.1. JDBC disadvantage

Object Relational Mapping (ORM) is designed to cover the JDBC disadvantages. There are three main reasons why using JDBC directly is not a good choice for many applications: [10]

1. Developing and maintaining SQL is difficult and time consuming
2. There is lack of portability with SQL
3. Writing JDBC code is time consuming and error prone

3.2. Hibernate

Hibernate is an object/relational mapping tool for Java environments. The term of Object Relational Mapping (ORM) refers to technique of mapping a data representation from an object model to a relational data model with a SQL-based schema. Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities and can significantly reduce development time otherwise spent with manual data handling in SQL and JDBC [11].

Hibernate is a popular Java ORM framework that is used by almost all java programmer. Hibernate try to relieve the developer from 95 percent of common data persistence related programming tasks. By using hibernate, developer can easily maintain the relation between object class and table. It tries to remove vendor-specified SQL code and help you with the common task of result set translation from tabular representation to graph of objects.

Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities and can significantly reduce development time otherwise spent with manual data handling in SQL and JDBC [6].

The characteristics of hibernate usage are: hibernate configuration, mapping object to it's table and relation. Hibernate has integration to many popular Java framework.

The architecture of hibernate can be described as the following:

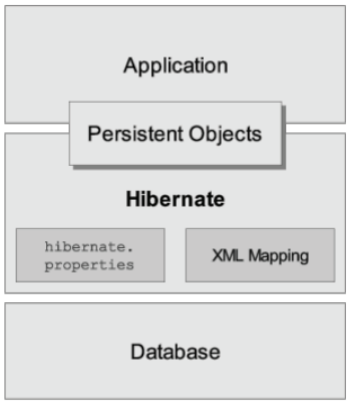

Figure 2 Hibernate Architecture [11]

The powerful of hibernate also imposes limitation, such as: (**the reason on why HCE is developed**)

- In many tables relationship, let hibernate select the data without projection will cause

unnecessary join, this will impact to performance. This is known as n+1 hibernate problem

- Projection has an association (e.g: "classB.name") will cause exception. Hibernate cannot return the value of "name" to property "classB" in the base class. (class to be fetched)

Projection contain association in hibernate still has complexity, we need to create criteria or alias to every join association.

IV. Hibernate Possible Improvement

ORM, especially hibernate, help us a lot in persistence data related programming tasks. We can easily insert data by only passing the object class and getting the data from database into collection of objects translated. Java programmer can save application development time. Hopefully, they will be happier to do their programming tasks.

Although hibernate gives us a significance benefit in persistence data layer, there are possible improvement to have hibernate more powerful. This paper strictly focused on hibernate improvement. Here are the possible improvements of hibernate:

- Easy projections

The default join fetch of hibernate is based on annotation or xml configuration. Projections with join need to call "createAlias" or "createCriteria" method. It will be very nice if we could do something like "secondTable.name" and the return value can be set to the secondTable field at the Java Object Class

- Easy Restrictions

When we need to search/filter of field in the second table, hibernate need to do join like point 1. It will be very nice if we could do something like "Restrictions.eq("seondTable.name", "[NAME]")" without calling the "createCriteria" or "createAlias" method

- Easy Order

Following the two points before, it will be very nice if we could do something like "Order.asc("secondTable.name")" to sort the data from database without call create criteria or alias method.

V. Solution Approach: Hibernate Criteria Extension (HCE)

To cover the possible improvement at the part 4, this research develops a "hibernate wrapper library" as the solution approach. This library is called as Hibernate Criteria Extension (HCE)

5.1 HCE Architecture

The following picture is the HCE Architecture:

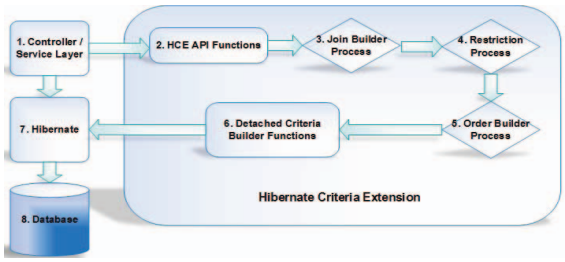

Figure 3 HCE Architecture

The architecture of HCE can be described as following:

- Application Controller/Service layer (1) will call HCE API functions (2) to retrieve query result, we can also pass this step by directly call hibernate function
- HCE will detect join from the projections given (3)
- After the join builder, HCE process the restriction (4)
- HCE will also process the order given (5)
- Return the detached Criteria object (6)
- Passing the detached Criteria object to hibernate
- Retrieve data from database and return to the controller/service caller method

5.2 HCE Features

The following are the HCE features:

- Declare hibernate projections by passing field(s) name, even an association. (select only the needed field(s)/column(s)).
- Declare restriction(s) by passing "Expression"; Expression contains: propertyName, restrictionType, value. This is valid for association.
- Declare order(s) by passing Order; Order contains: propertyName, orderType. This is valid for association.

5.3 HCE Usages Example

The example use of HCE will be discussed at this section. Suppose we have three tables like the following picture:

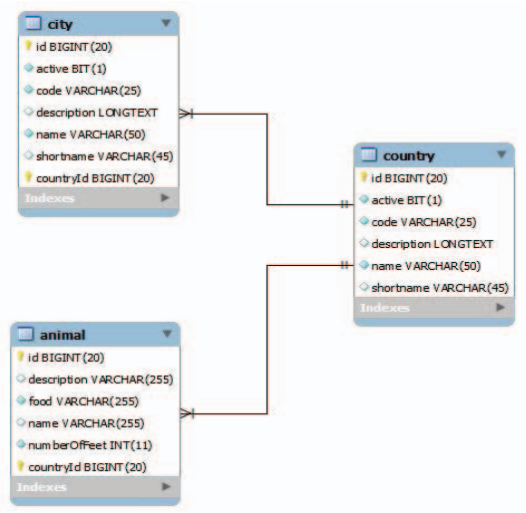

Figure 4 ERD for HCE Usage

The ERD has three tables that means we need to create three hibernate mapping objects. Animal has many-to-one relationship to country and country has one-to-many relationship to city.

HCE can be used to select data based on the relationship mapping in hibernate. For now, only annotation mapping is supported. HCE can detect join by using dot in projections and set the selected value to the property of associated class where hibernate cannot in term of using projections. Hibernate can do it if we set eager mode without projections.

The usages of HCE can be described at the following (using Figure 5 ERD Example):

TABLE I. HIBERNATE CRITERIA EXTENSION USAGES

Command	HCE Code
Select one column from a table	DetachedCriteria detachedCriteria = criteriaWithProjections(**new** String[] {"name"});
Select many columns from a table	DetachedCriteria detachedCriteria = criteriaWithProjections(new String[]{ "id", "name", "food", "numberOfFeet", "description"});
Join table (association)	DetachedCriteria detachedCriteria = criteriaWithProjections(new String[]{"id", "name", "country.name"});
Select columns and add restriction	DetachedCriteria detachedCriteria = criteriaByProperty(new String[]{"id", "name", "description"}, Expression.eq("name", "Panda"));
Select columns from a table and add restriction to another	DetachedCriteria detachedCriteria = criteriaByProperty(new String[]{"id", "name", "description"}, Expression.eq("country.name", "China"));
Select columns from two table and add restriction to the second table	DetachedCriteria detachedCriteria = criteriaByProperty(new String[]{"id", "name", "description", "country.name" }, Expression.eq("country.name", "China"));
Restrictions type	DetachedCriteria detachedCriteria = criteriaByProperty(null, Expression.like("name", "Panda"));
Order	DetachedCriteria detachedCriteria = criteriaWithProjections(null, softtech.hong.hce.model.Order.asc("name")) ;
Paging	DetachedCriteria[] detachedCriterias = queryTransformer(null, projections, orders); *null mean that no filter is applied
Query by Example	Animal animal = new Animal(); animal.setName("Panda"); DetachedCriteria detachedCriteria = queryTranslation(animal, null, null); *first null for projections, second one for ignore properties, in hibernate it's called exclude properties.
Query by Example on an association	Animal animal = new Animal(); animal.setName("Panda"); Country country = new Country(); country.setName("China"); animal.setCountry(country); DetachedCriteria detachedCriteria = queryTranslation(animal, null, null); *animal name and country name will be assigned as restrictions.

The benefits of HCE compared to hibernate:

- An association using hibernate criteria need to call createCriteria or createAlias method. HCE can detect association defined in projection only by using dot, such as: animal.country.city.name. Hibernate need createAlias to both country and city for getting the city name value

- Association restriction in hibernate criteria is also need create criteria or alias method. HCE can also perform restriction using dot, such as: Expression.eq("country.city.name","city-name")
- Order can also be performed by using dot

VI. HCE LIMITATION

HCE is used to cover commonly use of hibernate criteria, in some cases, HCE cannot be used as well as hibernate does. Here are the limitations of HCE:

- HCE only support hibernate annotation configuration, xml configuration is not supported as the last commonly use is annotation configuration

- Multilevel association, one-to-many and many-to- many hasn't been supported well. I suggest to let projections is null and just use the Expression. The example of multilevel association: Country has many City and City has many District

VII. FUTURE WORK

HCE supports projection by passing field's name of object Java, we can develop dynamic query where user can select the field and it's relation(s) to be shown. HCE will detect automatically the table's field(s) and their relations. This dynamic query feature can be used in reporting or the other systems.

VIII. CONCLUSION

HCE is an extension library to hibernate, not a hibernate substitution library, we can use it for simplifying query in criteria. For some cases described at part 5, hibernate knowledge is needed for understanding HCE. Although HCE can be used in many select cases, hibernate or JDBC is needed in some cases, such as: JDBC is a better choice for large data set query, multilevel bag collection is not supported yet by HCE, so hibernate is a choice.

REFERENCES

[1] F. Of and S. Edition, *Database*. .

[2] U. T. Nacional, "Development of a Relational Database Management System .," vol. 3, no. 2, pp. 33–37, 2003.

[3] P. Atzeni, Ceri, Paraboschi, and Torlone, "Database Systems."

[4] A. Lipitsäinen, "ORM – Object Relational Mapping," pp. 1–30.

[5] D. Systems, "Introduction to Database Concepts."

[6] E. Erhieyovwe, P. Oghenekaro, and N. Oluwole, "An Object Relational Mapping Technique for Java Framework," vol. 2, no. 6, pp. 1–9, 2013.

[7] J. Barnes, "Object-relational mapping as a persistence mechanism for object-oriented applications," p. 113, 2007.

[8] C. Ireland and D. Bowers, "Exposing the Myth : Object-Relational Impedance Mismatch is a Wicked Problem," no. c, pp. 21–26, 2015.

[9] C. Ireland, D. Bowers, M. Newton, and K. Waugh, "Understanding object-relational mapping: A framework based approach," vol. 2, no. 2, pp. 202–216, 2009.

[10] S. Lokhande, R. Patil, and A. Kadam, "Use of Hibernate in modern technology : Project Management," vol. 2, pp. 222–227, 2010.

[11] JBoss, "JBoss Enterprise Application Platform," no. 12/2009, 2009.