**Hibernate Query examples (HQL)**

Hibernate created a new language named Hibernate Query Language (HQL), the syntax is quite similar to database SQL language. The main difference between is **HQL uses class name instead of table name, and property names instead of column name**.

HQL is extremely simple to learn and use, and the code is always self-explanatory.

**1. HQL Select Query Example**

Retrieve a stock data where stock code is “7277”.

Query query = session.createQuery("from Stock where stockCode = :code ");

query.setParameter("code", "7277");

List list = query.list();

Query query = session.createQuery("from Stock where stockCode = '7277' ");

List list = query.list();

**2. HQL Update Query Example**

Update a stock name to “DIALOG1” where stock code is “7277”.

Query query = session.createQuery("update Stock set stockName = :stockName" +

" where stockCode = :stockCode");

query.setParameter("stockName", "DIALOG1");

query.setParameter("stockCode", "7277");

**int** result = query.executeUpdate();

Query query = session.createQuery("update Stock set stockName = 'DIALOG2'" +

" where stockCode = '7277'");

**int** result = query.executeUpdate();

**3. HQL Delete Query Example,** Delete a stock where stock code is “7277”.

Query query = session.createQuery("delete Stock where stockCode = :stockCode");

query.setParameter("stockCode", "7277");

**int** result = query.executeUpdate();

Query query = session.createQuery("delete Stock where stockCode = '7277'");

**int** result = query.executeUpdate();

**4. HQL Insert Query Example**

In HQL, only the INSERT INTO … SELECT … is supported; there is no INSERT INTO … VALUES. HQL only support insert from another table. For example

"insert into Object (id, name) select oo.id, oo.name from OtherObject oo";

Insert a stock record from another backup\_stock table. This can also called bulk-insert statement.

Query query = session.createQuery("insert into Stock(stock\_code, stock\_name)" +

"select stock\_code, stock\_name from backup\_stock");

**int** result = query.executeUpdate();

The **query.executeUpdate()** will return how many number of record has been inserted, updated or deleted.

# Hibernate parameter binding examples

Without parameter binding, you have to concatenate the parameter String like this (bad code) :

String hql = "from Stock s where s.stockCode = '" + stockCode + "'";

List result = session.createQuery(hql).list();

Pass an unchecked value from user input to the database will raise security concern, because it can easy get hack by SQL injection. You have to avoid the above bad code and using parameter binding instead.

**Hibernate parameter binding,** There are two ways to parameter binding : named parameters or positional.

**1. Named parameters**

This is the most common and user friendly way. It use colon followed by a parameter name (:example) to define a named parameter. See examples…

##### **Example 1 – setParameter**

The **setParameter** is smart enough to discover the parameter data type for you.

String hql = "from Stock s where s.stockCode = :stockCode";

List result = session.createQuery(hql)

.setParameter("stockCode", "7277")

.list();

##### **Example 2 – setString**

You can use **setString** to tell Hibernate this parameter date type is String.

String hql = "from Stock s where s.stockCode = :stockCode";

List result = session.createQuery(hql)

.setString("stockCode", "7277")

.list();

##### **Example 3 – setProperties**

This feature is great ! You can pass an object into the parameter binding. Hibernate will automatic check the object’s properties and match with the colon parameter.

Stock stock = **new** Stock();

stock.setStockCode("7277");

String hql = "from Stock s where s.stockCode = :stockCode";

List result = session.createQuery(hql)

.setProperties(stock)

.list();

**2. Positional parameters**

It’s use question mark (?) to define a named parameter, and you have to set your parameter according to the position sequence. See example…

String hql = "from Stock s where s.stockCode = ? and s.stockName = ?";

List result = session.createQuery(hql)

.setString(0, "7277")

.setParameter(1, "DIALOG")

.list();

This approach is not support the **setProperties** function. In addition, it’s vulnerable to easy breakage because every change of the position of the bind parameters requires a change to the parameter binding code.

String hql = "from Stock s where s.stockName = ? and s.stockCode = ?";

List result = session.createQuery(hql)

.setParameter(0, "DIALOG")

.setString(1, "7277")

.list();

## Conclusion

In Hibernate parameter binding, i would recommend always go for “**Named parameters**“, as it’s more easy to maintain, and the compiled SQL statement can be reuse (if only bind parameters change) to increase the performance.

# How to embed Oracle hints in Hibernate query

With Oracle hints, you can alter the Oracle execution plans to affect the way how Oracle retrieve the data from database. Go here for more detail about [Oracle optimizer hints](http://download.oracle.com/docs/cd/B10501_01/server.920/a96533/hintsref.htm).

In Hibernate, is this possible to embed the Oracle hint into the Hibernate query?

**Hibernate setComment()?**

Can you embed the Oracle hint into HQL with Hibernate custom comment “**setComment()**” function? Let’s see an example here

**1. Original Hibernate Query** This is a simple select HQL to retrieve a Stock with a stock code.

String hql = "from Stock s where s.stockCode = :stockCode";

List result = session.createQuery(hql)

.setString("stockCode", "7277")

.list();

Hibernate:

select

stock0\_.STOCK\_ID as STOCK1\_0\_,

stock0\_.STOCK\_CODE as STOCK2\_0\_,

stock0\_.STOCK\_NAME as STOCK3\_0\_

from stock stock0\_

where stock0\_.STOCK\_CODE=?

**2. Try Hibernate setComment()**

Enable the **hibernate.use\_sql\_comments** in Hibernate’s configuration file (hibernate.cfg.xml) in order to output the custom comment to your log file or console.

<hibernate-configuration>

<session-factory>

...

<property name="show\_sql">**true**</property>

<property name="format\_sql">**true**</property>

<property name="use\_sql\_comments">**true**</property>

<mapping **class**=" Stock" />

</session-factory>

</hibernate-configuration>

Using Hibernate **setComment()** to insert a custom comment to your query.

String hql = "from Stock s where s.stockCode = :stockCode";

List result = session.createQuery(hql)

.setString("stockCode", "7277")

.setComment("+ INDEX(stock idx\_stock\_code)")

.list();

Hibernate:

/\* + INDEX(stock idx\_stock\_code) \*/ select

stock0\_.STOCK\_ID as STOCK1\_0\_,

stock0\_.STOCK\_CODE as STOCK2\_0\_,

stock0\_.STOCK\_NAME as STOCK3\_0\_

from stock stock0\_

where stock0\_.STOCK\_CODE=?

**3. Is this work?**

It’s not, there are two problem with Hibernate custom comments.

1. The Oracle hint have to append after the ‘select’, not before.

Hibernate generated query

/\* + INDEX(stock idx\_stock\_code) \*/ select

The correct way should be…

select /\*+ INDEX(stock idx\_stock\_code) \*/

2. Hibernate will add an extra space in between “/\* +” automatically.

In Hibernate, there are still no official way to embed the Oracle hints into Hibernate query langueges (HQL).

P.S Thanks Pete contribute on this.

**Working solution**

The only solution is using the Hibernate **createSQLQuery** method to execute the native SQL statemen

String hql = "/\*+ INDEX(stock idx\_stock\_code) \*/

select \* from stock s where s.stock\_code = :stockCode";

List result = session.createQuery(hql)

.setString("stockCode", "7277")

.list();

Hibernate:

/\*+ INDEX(stock idx\_stock\_code) \*/ select \*

from stock s where s.stock\_code = ?