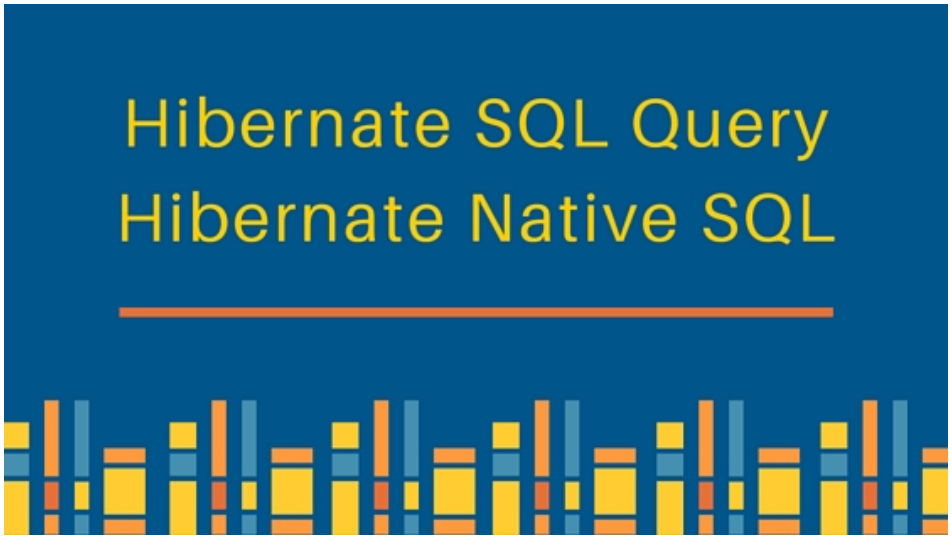
Hibernate Native SQL Query Example

APRIL 2, 2018 BY PANKAJ — 8 COMMENTS

Welcome to the Hibernate Native SQL Query example tutorial. We looked into Hibernate Query Language and Hibernate Criteria in earlier articles, today we will look into Hibernate Native SQL query with examples.

Table of Contents [hide]

Hibernate SQL Query



Hibernate provide option to execute native SQL queries through the use of **SQLQuery** object. Hibernate SQL Query is very handy when we have to execute database vendor specific queries that are not supported by Hibernate API. For example query hints or the CONNECT keyword in Oracle Database.

For normal scenarios, Hibernate SQL query is not the recommended approach because we loose benefits related to hibernate association and hibernate first level cache.

I will use MySQL database and same tables and data setup as used in HQL example, so you should check out that first to understand the tables and corresponding model classes mapping.

Hibernate Native SQL Example

For Hibernate Native SQL Query, we use `Session.createSQLQuery(String query)` to create the SQLQuery object and execute it. For example, if you want to read all the records from Employee table, we can do it through below code.

```
// Prep work
SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
Session session = sessionFactory.getCurrentSession();

// Get All Employees
Transaction tx = session.beginTransaction();
SQLQuery query = session.createSQLQuery("select emp_id, emp_name, emp_salary from
Employee");
List<Object[]> rows = query.list();
for(Object[] row : rows){
        Employee emp = new Employee();
        emp.setId(Long.parseLong(row[0].toString()));
        emp.setName(row[1].toString());
        emp.setSalary(Double.parseDouble(row[2].toString()));
```

```
        System.out.println(emp);
}
```

When we execute above code for the data setup we have, it produces following output.

```
Hibernate: select emp_id, emp_name, emp_salary from Employee
Id= 1, Name= Pankaj, Salary= 100.0, {Address= null}
Id= 2, Name= David, Salary= 200.0, {Address= null}
Id= 3, Name= Lisa, Salary= 300.0, {Address= null}
Id= 4, Name= Jack, Salary= 400.0, {Address= null}
```

Notice that `list()` method returns the List of Object array, we need to explicitly parse them to double, long etc. Our Employee and Address classes have following `toString()` method implementations.

```
@Override
public String toString() {
        return "Id= " + id + ", Name= " + name + ", Salary= " + salary
                        + ", {Address= " + address + "}";
}
```

```
@Override
public String toString() {
        return "AddressLine1= " + addressLine1 + ", City=" + city
                        + ", Zipcode=" + zipcode;
}
```

Notice that our query is not returning Address data, whereas if we use HQL query `"from Employee"`, it returns the associated table data too.

Hibernate SQL Query addScalar

Hibernate uses `ResultSetMetadata` to deduce the type of the columns returned by the query, from performance point of view we can use `addScalar()` method to define the data type of the column. However we would still get the data in form of Object array.

```
//Get All Employees - addScalar example
query = session.createSQLQuery("select emp_id, emp_name, emp_salary from Employee")
                .addScalar("emp_id", new LongType())
                .addScalar("emp_name", new StringType())
                .addScalar("emp_salary", new DoubleType());
```

```
rows = query.list();
for(Object[] row : rows){
        Employee emp = new Employee();
        emp.setId(Long.parseLong(row[0].toString()));
        emp.setName(row[1].toString());
        emp.setSalary(Double.parseDouble(row[2].toString()));
        System.out.println(emp);
}
```

The output generated will be same, however we will see slight performance improvement when the data is huge.

Hibernate Native SQL Multiple Tables

If we would like to get data from both Employee and Address tables, we can simply write the SQL query for that and parse the result set.

```
query = session.createSQLQuery("select e.emp_id, emp_name, emp_salary,address_line1,
city,
        zipcode from Employee e, Address a where a.emp_id=e.emp_id");
rows = query.list();
for(Object[] row : rows){
        Employee emp = new Employee();
        emp.setId(Long.parseLong(row[0].toString()));
        emp.setName(row[1].toString());
        emp.setSalary(Double.parseDouble(row[2].toString()));
        Address address = new Address();
        address.setAddressLine1(row[3].toString());
        address.setCity(row[4].toString());
        address.setZipcode(row[5].toString());
        emp.setAddress(address);
        System.out.println(emp);
}
```

For above code, the output produced will be like below.

```
Hibernate: select e.emp_id, emp_name, emp_salary,address_line1, city, zipcode from
Employee e, Address a where a.emp_id=e.emp_id
```

```
Id= 1, Name= Pankaj, Salary= 100.0, {Address= AddressLine1= Albany Dr, City=San Jose, Zipcode=95129}
Id= 2, Name= David, Salary= 200.0, {Address= AddressLine1= Arques Ave, City=Santa Clara, Zipcode=95051}
Id= 3, Name= Lisa, Salary= 300.0, {Address= AddressLine1= BTM 1st Stage, City=Bangalore, Zipcode=560100}
Id= 4, Name= Jack, Salary= 400.0, {Address= AddressLine1= City Centre, City=New Delhi, Zipcode=100100}
```

Hibernate Native SQL Entity and Join

We can also use `addEntity()` and `addJoin()` methods to fetch the data from associated table using tables join. For example, above data can also be retrieved as below.

```
//Join example with addEntity and addJoin
query = session.createSQLQuery("select {e.*}, {a.*} from Employee e join Address a ON e.emp_id=a.emp_id")
                .addEntity("e",Employee.class)
                .addJoin("a","e.address");
rows = query.list();
for (Object[] row : rows) {
    for(Object obj : row) {
        System.out.print(obj + "::");
    }
    System.out.println("\n");
}
//Above join returns both Employee and Address Objects in the array
for (Object[] row : rows) {
        Employee e = (Employee) row[0];
        System.out.println("Employee Info::"+e);
        Address a = (Address) row[1];
        System.out.println("Address Info::"+a);
}
```

`{[aliasname].*}` is used to return all properties of an entity. When we use `addEntity()` and `addJoin()` with join queries like above it returns both the objects, as shown above.

Output produced by above code is like below.

```
Hibernate: select e.emp_id as emp_id1_1_0_, e.emp_name as emp_name2_1_0_,
e.emp_salary as emp_sala3_1_0_, a.emp_id as emp_id1_0_1_, a.address_line1 as
address_2_0_1_, a.city as city3_0_1_, a.zipcode as zipcode4_0_1_ from Employee e join
```

```
Address a ON e.emp_id=a.emp_id
Id= 1, Name= Pankaj, Salary= 100.0, {Address= AddressLine1= Albany Dr, City=San Jose,
Zipcode=95129}::AddressLine1= Albany Dr, City=San Jose, Zipcode=95129::

Id= 2, Name= David, Salary= 200.0, {Address= AddressLine1= Arques Ave, City=Santa
Clara, Zipcode=95051}::AddressLine1= Arques Ave, City=Santa Clara, Zipcode=95051::

Id= 3, Name= Lisa, Salary= 300.0, {Address= AddressLine1= BTM 1st Stage,
City=Bangalore, Zipcode=560100}::AddressLine1= BTM 1st Stage, City=Bangalore,
Zipcode=560100::

Id= 4, Name= Jack, Salary= 400.0, {Address= AddressLine1= City Centre, City=New
Delhi, Zipcode=100100}::AddressLine1= City Centre, City=New Delhi, Zipcode=100100::

Employee Info::Id= 1, Name= Pankaj, Salary= 100.0, {Address= AddressLine1= Albany Dr,
City=San Jose, Zipcode=95129}
Address Info::AddressLine1= Albany Dr, City=San Jose, Zipcode=95129
Employee Info::Id= 2, Name= David, Salary= 200.0, {Address= AddressLine1= Arques Ave,
City=Santa Clara, Zipcode=95051}
```

You can run both the queries in the mysql client and notice that the output produced is same.

```
mysql> select e.emp_id as emp_id1_1_0_, e.emp_name as emp_name2_1_0_, e.emp_salary as
emp_sala3_1_0_, a.emp_id as emp_id1_0_1_, a.address_line1 as address_2_0_1_, a.city
as city3_0_1_, a.zipcode as zipcode4_0_1_ from Employee e join Address a ON
e.emp_id=a.emp_id;
+--------------+----------------+----------------+--------------+----------------+---
----------+---------------+
| emp_id1_1_0_ | emp_name2_1_0_ | emp_sala3_1_0_ | emp_id1_0_1_ | address_2_0_1_ |
city3_0_1_   | zipcode4_0_1_ |
+--------------+----------------+----------------+--------------+----------------+---
----------+---------------+
|            1 | Pankaj         |            100 |            1 | Albany Dr      |
San Jose     | 95129         |
|            2 | David          |            200 |            2 | Arques Ave     |
Santa Clara  | 95051         |
|            3 | Lisa           |            300 |            3 | BTM 1st Stage  |
Bangalore    | 560100        |
|            4 | Jack           |            400 |            4 | City Centre    |
New Delhi    | 100100        |
+--------------+----------------+----------------+--------------+----------------+---
----------+---------------+
4 rows in set (0.00 sec)
```

Hibernate Native SQL Query with Parameters

We can also pass parameters to the Hibernate SQL queries, just like JDBC PreparedStatement. The parameters can be set using the name as well as index, as shown in below example.

```
query = session
                .createSQLQuery("select emp_id, emp_name, emp_salary from Employee
where emp_id = ?");
List<Object[]> empData = query.setLong(0, 1L).list();
for (Object[] row : empData) {
        Employee emp = new Employee();
        emp.setId(Long.parseLong(row[0].toString()));
        emp.setName(row[1].toString());
        emp.setSalary(Double.parseDouble(row[2].toString()));
        System.out.println(emp);
}

query = session
                .createSQLQuery("select emp_id, emp_name, emp_salary from Employee
where emp_id = :id");
empData = query.setLong("id", 2L).list();
for (Object[] row : empData) {
        Employee emp = new Employee();
        emp.setId(Long.parseLong(row[0].toString()));
        emp.setName(row[1].toString());
        emp.setSalary(Double.parseDouble(row[2].toString()));
        System.out.println(emp);
```

Output produced by above code would be:

```
Hibernate: select emp_id, emp_name, emp_salary from Employee where emp_id = ?
Id= 1, Name= Pankaj, Salary= 100.0, {Address= null}
Hibernate: select emp_id, emp_name, emp_salary from Employee where emp_id = ?
Id= 2, Name= David, Salary= 200.0, {Address= null}
```

That's all for a brief introduction of Hibernate SQL Query, you should avoid using it unless you want to execute any database specific queries.

« PREVIOUS
Hibernate Log4j Logging

NEXT »
Hibernate Named Query Example – @NamedQuery

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: HIBERNATE

Comments

elon says
DECEMBER 26, 2017 AT 1:01 AM
Thanks u.

Reply

sarath says
MAY 19, 2016 AT 5:56 AM
Hi pankaj,
im sarath,nic explnation and good examples,and i have a problem with my coding when i use createSQLquery in session factory for executing a native query. it throws an exception like no dialect mapping for jdbc type 1111.
im using springs and hibernate and my database is db2, pls giv me a solution for this.
Reply

Anky says
JULY 15, 2016 AT 2:23 AM
You can set dialect for the particular database in hibernate configuration that may solve your issue.
Reply

Maurice says
APRIL 22, 2016 AT 6:59 AM
You saved my day! Thanks for the clear examples!
Reply

Kiran says
DECEMBER 15, 2015 AT 9:39 PM
How we use subquery like- select name ,(select state_name from Mst_State where stateid=1) from Mst_emp where emp_code =1000; data show in jsp
Reply

sathish says
NOVEMBER 23, 2015 AT 11:37 PM
Hello All,
How can I call the stored procedure which is having out perimeter from hibernate native sql queries?
Thanks in advance.
Reply

Supraja says
MARCH 10, 2015 AT 5:28 AM
cant we use iterate() to retrieve records from databases while using Native SQL Query??
Reply

Karthikeyan says
DECEMBER 8, 2014 AT 4:51 PM
Thanks for Shareing , it really helps me to understand better
Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

- Hibernate Spring MVC
- Hibernate Struts 2
- Hibernate Primefaces
- Hibernate Primefaces Spring
- Hibernate SpringRoo Primefaces
- Hibernate JSF Spring

Miscellaneous

- Hibernate Tools Eclipse Plugin
- Hibernate Configuration Offline
- [Solved] No identifier specified
- Hibernate Program Not Terminating
- Access to DialectResolutionInfo
- get is not valid
- No CurrentSessionContext configured
- Hibernate Interview Questions

RECOMMENDED TUTORIALS

Java Tutorials

- Java IO
- Java Regular Expressions
- Multithreading in Java
- Java Logging
- Java Annotations
- Java XML
- Collections in Java
- Java Generics
- Exception Handling in Java
- Java Reflection
- Java Design Patterns
- JDBC Tutorial

Java EE Tutorials

- Servlet JSP Tutorial
- Struts2 Tutorial
- Spring Tutorial
- Hibernate Tutorial
- Primefaces Tutorial
- Apache Axis 2
- JAX-RS
- Memcached Tutorial