Design By:Anil Kumar

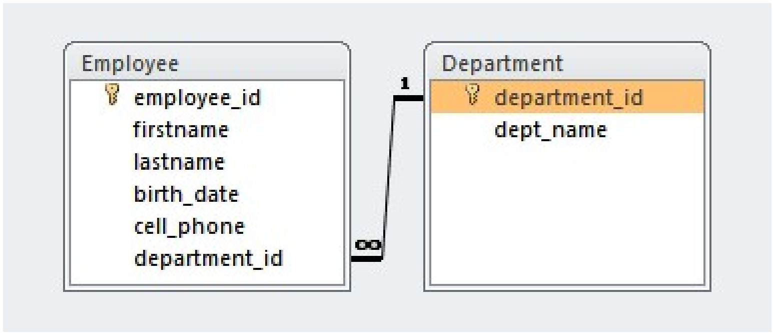
Hibernate Fetching Strategy

There are four fetching strategies

1. fetch-“join” = Disable the lazy loading, always load all the collections and entities.
2. fetch-“select” (default) = Lazy load all the collections and entities.
3. batch-size=”N” = Fetching up to ‘N’ collections or entities, *Not record*.
4. fetch-“subselect” = Group its collection into a sub select statement.

Fetching strategies examples

Here’s a “one-to-many relationship” example for the fetching strategies demonstration, a department is belong to many employees.



Example to declare fetch strategies in annotation

```
@Entity
@Table(name="department")
public class Department{
          . . . . .

        @OneToMany(mappedBy="department",
                cascade=CascadeType.ALL,fetch=FetchType.LAZY)
        @Fetch(FetchMode.SELECT)

        private Set<Employee> employee=new HashSet<Employee>();
        . . . . .
}
```

```
@Fetch(FetchMode.SELECT)
```

```
@Fetch(FetchMode.JOIN)
```

```
@BatchSize(size=5)
```

```
@Fetch(FetchMode.SUBSELECT)
```

1. fetch=”select” or @Fetch(FetchMode.SELECT)

This is the default fetching strategy. it enabled the lazy loading of all it’s related collections.

```
            long dptid=1;
            Department dept=(Department)session.get(Department.class, dptid);
            Set<Employee> employees=dept.getEmployee();

            /*Iterator<Employee> it=employees.iterator();

            while(it.hasNext()){
                  Employee e=it.next();
                  System.out.println(e.getEmpfirstname());
            }*/
```

OUTPUT: first select record from parent table only.

```
Hibernate: select department0_.department_id as department1_0_0_,
department0_.dept_name as dept2_0_0_ from department department0_ where
department0_.department_id=?
```

NOW:

```
            Iterator<Employee> it=employees.iterator();

            while(it.hasNext()){
                  Employee e=it.next();
                  System.out.println(e.getEmpfirstname());
            }
```

OUTPUT: Now select record from child table when it call.

```
Hibernate: select employee0_.department_id as department6_0_3_,
employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_,
employee0_.birth_date as birth2_1_2_, employee0_.cell_phone as cell3_1_2_,
employee0_.department_id as department6_1_2_, employee0_.firstname as firstname1_2_,
employee0_.lastname as lastname1_2_, employeede1_.employee_id as employee1_2_0_,
employeede1_.city as city2_0_, employeede1_.country as country2_0_,
employeede1_.state as state2_0_, employeede1_.street as street2_0_,
employee2_.employee_id as employee1_1_1_, employee2_.birth_date as birth2_1_1_,
employee2_.cell_phone as cell3_1_1_, employee2_.department_id as department6_1_1_,
employee2_.firstname as firstname1_1_, employee2_.lastname as lastname1_1_ from
employee employee0_ left outer join employeedetail employeede1_ on
employee0_.employee_id=employeede1_.employee_id left outer join employee employee2_
on employeede1_.employee_id=employee2_.employee_id where employee0_.department_id=?
```

2. fetch=”join” or @Fetch(FetchMode.JOIN)

The “join” fetching strategy will disabled the lazy loading of all it’s related collections.

```
long dptid=1;
Department dept=(Department)session.get(Department.class, dptid);
Set<Employee> employees=dept.getEmployee();

Iterator<Employee> it=employees.iterator();

while(it.hasNext()){
      Employee e=it.next();
      System.out.println(e.getEmpfirstname());
}
```

```
OUTPUT: Only single query fired for fetch all records no lazy calls.
```

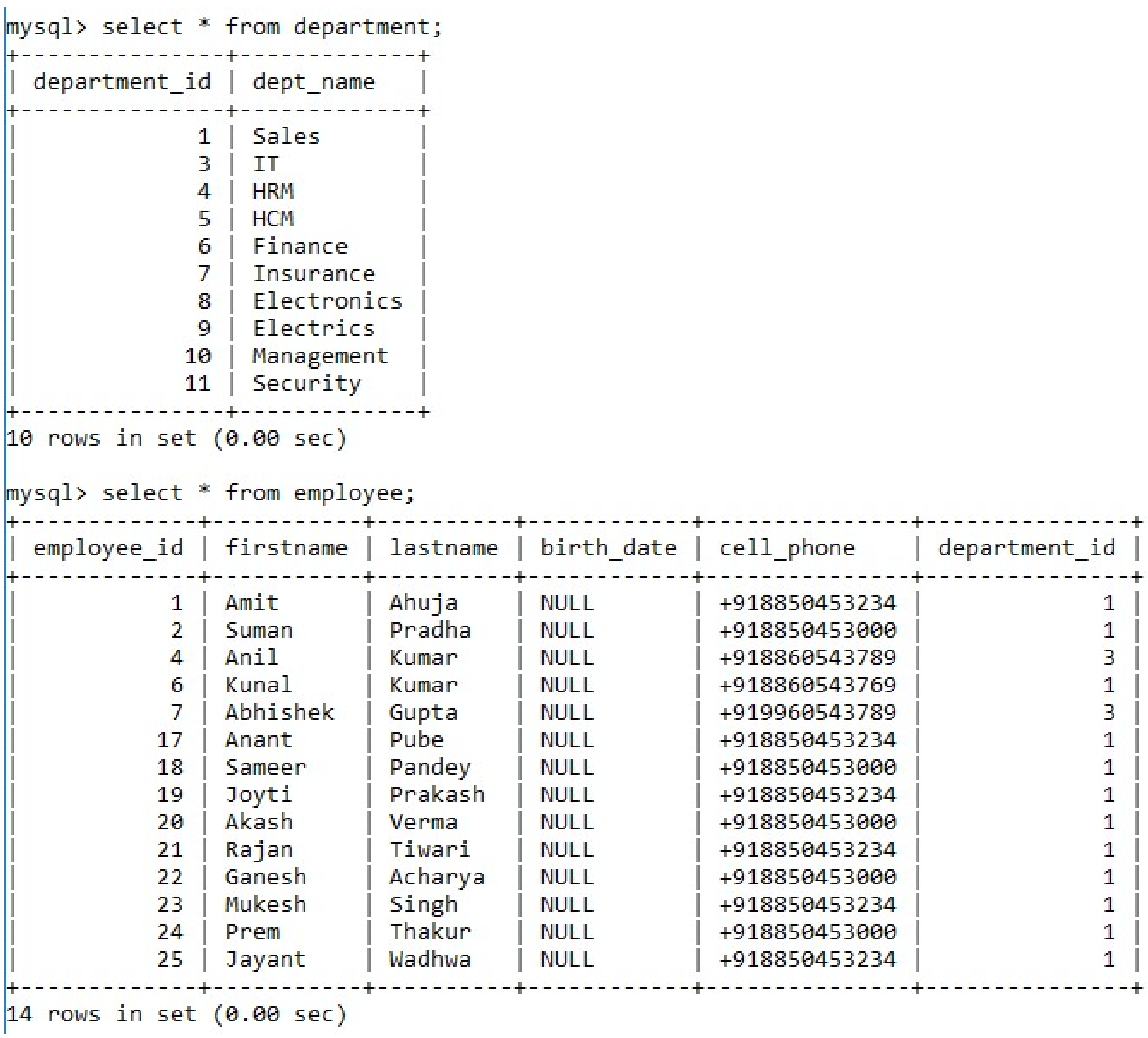
```
Hibernate: select department0_.department_id as department1_0_2_,
department0_.dept_name as dept2_0_2_, employee1_.department_id as
department6_0_4_, employee1_.employee_id as employee1_4_,
employee1_.employee_id as employee1_1_0_, employee1_.birth_date as
birth2_1_0_, employee1_.cell_phone as cell3_1_0_, employee1_.department_id as
department6_1_0_, employee1_.firstname as firstname1_0_, employee1_.lastname
as lastname1_0_, employeede2_.employee_id as employee1_2_1_,
employeede2_.city as city2_1_, employeede2_.country as country2_1_,
employeede2_.state as state2_1_, employeede2_.street as street2_1_ from
department department0_ left outer join employee employee1_ on
department0_.department_id=employee1_.department_id left outer join
employeedetail employeede2_ on
employee1_.employee_id=employeede2_.employee_id where
department0_.department_id=?
```

3. batch-size=”5” or @BatchSize(size = 5)

The batch-size fetching strategy is not define how many records inside in the collections are loaded. Instead, it defines how many collections should be loaded.

Let see another example, you want to print out all the department records and its related employee records (collections) one by one.

First we will see the db records:

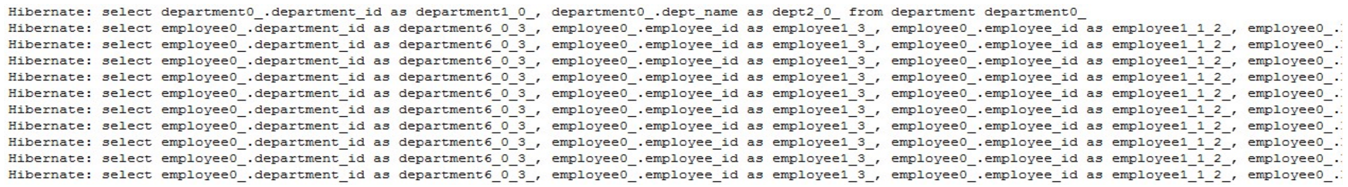


```
mysql> select * from department;
+---------------+-------------+
| department_id | dept_name   |
+---------------+-------------+
|             1 | Sales       |
|             3 | IT          |
|             4 | HRM         |
|             5 | HCM         |
|             6 | Finance     |
|             7 | Insurance   |
|             8 | Electronics |
|             9 | Electrics   |
|            10 | Management  |
|            11 | Security    |
+---------------+-------------+
10 rows in set (0.00 sec)

mysql> select * from employee;
+-------------+-----------+----------+------------+---------------+---------------+
| employee_id | firstname | lastname | birth_date | cell_phone    | department_id |
+-------------+-----------+----------+------------+---------------+---------------+
|           1 | Amit      | Ahuja    | NULL       | +918850453234 |             1 |
|           2 | Suman     | Pradha   | NULL       | +918850453000 |             1 |
|           4 | Anil      | Kumar    | NULL       | +918860543789 |             3 |
|           6 | Kunal     | Kumar    | NULL       | +918860543769 |             1 |
|           7 | Abhishek  | Gupta    | NULL       | +919960543789 |             3 |
|          17 | Anant     | Pube     | NULL       | +918850453234 |             1 |
|          18 | Sameer    | Pandey   | NULL       | +918850453000 |             1 |
|          19 | Joyti     | Prakash  | NULL       | +918850453234 |             1 |
|          20 | Akash     | Verma    | NULL       | +918850453000 |             1 |
|          21 | Rajan     | Tiwari   | NULL       | +918850453234 |             1 |
|          22 | Ganesh    | Acharya  | NULL       | +918850453000 |             1 |
|          23 | Mukesh    | Singh    | NULL       | +918850453234 |             1 |
|          24 | Prem      | Thakur   | NULL       | +918850453000 |             1 |
|          25 | Jayant    | Wadhwa   | NULL       | +918850453234 |             1 |
+-------------+-----------+----------+------------+---------------+---------------+
14 rows in set (0.00 sec)
```

Here total 10 rows for department and 14 rows for employee;

```java
List<Department> deptlist=session.createQuery("from Department").list();
for(Department d:deptlist){
            Set<Employee> employees=d.getEmployee();
            Iterator<Employee> it=employees.iterator();

            while(it.hasNext()){
                  Employee e=it.next();
                  System.out.println(e.getEmpfirstname());
            }
}
```

OUTPUT: without BatchSize (Total 10 select query fired for fetch all records)



```
Hibernate: select department0_.department_id as department1_0_, department0_.dept_name as dept2_0_ from department department0_
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
Hibernate: select employee0_.department_id as department6_0_3_, employee0_.employee_id as employee1_3_, employee0_.employee_id as employee1_1_2_, employee0_.
```

OUTPUT: With BatchSize=5 then two select query fired set of five department id per set.

```
Hibernate: select department0_.department_id as department1_0_,
department0_.dept_name as dept2_0_ from department department0_
```

```
Hibernate: select employee0_.department_id as department6_0_3_,
employee0_.employee_id as employee1_3_, employee0_.employee_id as
employee1_1_2_, employee0_.birth_date as birth2_1_2_, employee0_.cell_phone
as cell3_1_2_, employee0_.department_id as department6_1_2_,
employee0_.firstname as firstname1_2_, employee0_.lastname as lastname1_2_,
employeede1_.employee_id as employee1_2_0_, employeede1_.city as city2_0_,
employeede1_.country as country2_0_, employeede1_.state as state2_0_,
employeede1_.street as street2_0_, employee2_.employee_id as employee1_1_1_,
employee2_.birth_date as birth2_1_1_, employee2_.cell_phone as cell3_1_1_,
employee2_.department_id as department6_1_1_, employee2_.firstname as
firstname1_1_, employee2_.lastname as lastname1_1_ from employee employee0_
left outer join employeedetail employeede1_ on
employee0_.employee_id=employeede1_.employee_id left outer join employee
employee2_ on employeede1_.employee_id=employee2_.employee_id where
employee0_.department_id in (?, ?, ?, ?, ?)
```

```
Hibernate: select employee0_.department_id as department6_0_3_,
employee0_.employee_id as employee1_3_, employee0_.employee_id as
employee1_1_2_, employee0_.birth_date as birth2_1_2_, employee0_.cell_phone
as cell3_1_2_, employee0_.department_id as department6_1_2_,
employee0_.firstname as firstname1_2_, employee0_.lastname as lastname1_2_,
employeede1_.employee_id as employee1_2_0_, employeede1_.city as city2_0_,
employeede1_.country as country2_0_, employeede1_.state as state2_0_,
employeede1_.street as street2_0_, employee2_.employee_id as employee1_1_1_,
employee2_.birth_date as birth2_1_1_, employee2_.cell_phone as cell3_1_1_,
employee2_.department_id as department6_1_1_, employee2_.firstname as
firstname1_1_, employee2_.lastname as lastname1_1_ from employee employee0_
left outer join employeedetail employeede1_ on
employee0_.employee_id=employeede1_.employee_id left outer join employee
employee2_ on employeede1_.employee_id=employee2_.employee_id where
employee0_.department_id in (?, ?, ?, ?, ?)
```

4. fetch=”subselect” or @Fetch(FetchMode.SUBSELECT)

```
List<Department> deptlist=session.createQuery("from Department").list();
for(Department d:deptlist){

      Set<Employee> employees=d.getEmployee();
      Iterator<Employee> it=employees.iterator();

      while(it.hasNext()){
      Employee e=it.next();
            System.out.println(e.getEmpfirstname());
      }
}
```

OUTPUT: Only single select query fired to get all records

```
Hibernate: select department0_.department_id as department1_0_,
department0_.dept_name as dept2_0_ from department department0_
```

```
Hibernate: select employee0_.department_id as department6_0_3_,
employee0_.employee_id as employee1_3_, employee0_.employee_id as
employee1_1_2_, employee0_.birth_date as birth2_1_2_, employee0_.cell_phone
as cell3_1_2_, employee0_.department_id as department6_1_2_,
employee0_.firstname as firstname1_2_, employee0_.lastname as lastname1_2_,
employeede1_.employee_id as employee1_2_0_, employeede1_.city as city2_0_,
employeede1_.country as country2_0_, employeede1_.state as state2_0_,
employeede1_.street as street2_0_, employee2_.employee_id as employee1_1_1_,
employee2_.birth_date as birth2_1_1_, employee2_.cell_phone as cell3_1_1_,
employee2_.department_id as department6_1_1_, employee2_.firstname as
firstname1_1_, employee2_.lastname as lastname1_1_ from employee employee0_
left outer join employeedetail employeede1_ on
employee0_.employee_id=employeede1_.employee_id left outer join employee
employee2_ on employeede1_.employee_id=employee2_.employee_id where
employee0_.department_id in (select department0_.department_id from
department department0_)
```

Difference between openSession and getCurrentSession

openSession

When you call SessionFactory.openSession, it always create new Session object afresh and give it to you. You need to explicitly flush and close these session objects. As session objects are not thread safe, you need to create one session object per request in multithreaded environment and one session per request in web applications too.

getCurrentSession

When you call SessionFactory. getCurrentSession, it will provide you session object which is in hibernate context and managed by hibernate internally. It is bound to transaction scope.

When you call SessionFactory. getCurrentSession, it creates a new Session if not exists , else use same session which is in current hibernate context. It automatically flush and close session when transaction ends, so you do not need to do externally.

If you are using hibernate in single threaded environment, you can use getCurrentSession, as it is faster in performance as compare to creating new session each time.

You need to add following property to hibernate.cfg.xml to use getCurrentSession method.

```
<session-factory>
<!--  Put other elements here -->
<property name="hibernate.current_session_context_class">
          thread
</property>
</session-factory>
```