hibernate缓存机制详细分析 - xiaoluo501395377 - 博客园

复制代码

您可以通过点击 **右下角** 的按钮 来对文章内容作出评价，也可以通过左下方的 **关注按钮** 来关注我的博客的最新动态。 如果文章内容对您有帮助，不要忘记点击右下角的 **推荐按钮** 来支持一下哦 如果您对文章内容有任何疑问，可以通过评论或发邮件的方式联系我：501395377@qq.com / lzp501395377@gmail.com如果需要转载，请注明出处，谢谢！！

复制代码

在本篇随笔里将会分析一下hibernate的缓存机制，包括一级缓存(session级别)、二级缓存(sessionFactory级别)以及查询缓存，当然还要讨论下我们的N+1的问题。

随笔虽长，但我相信看完的朋友绝对能对hibernate的 N+1问题以及缓存有更深的了解。

一、N+1问题

首先我们来探讨一下N+1的问题，我们先通过一个例子来看一下，什么是N+1问题：

list()获得对象：

复制代码

```
/**  * 此时会发出一条sql，将30个学生全部查询出来  */  List ls = (List)session.createQuery("from Student")  .setFirstResult(0).setMaxResults(30).list(); Iterator stus = ls.iterator();  for(;stus.hasNext();)  {  Student stu = (Student)stus.next();  System.out.println(stu.getName());  }
```

复制代码

如果通过list()方法来获得对象，毫无疑问，hibernate会发出一条sql语句，将所有的对象查询出来，这点相信大家都能理解

```
Hibernate: select student0_.id as id2_, student0_.name as name2_, student0_.rid as rid2_, student0_.sex as sex2_ from t_student student0_ limit ?
```

那么，我们再来看看iterator()这种情况

iterator()获得对象

复制代码

```
/**  * 如果使用iterator方法返回列表，对于hibernate而言，它仅仅只是发出取id列表的sql * 在查询相应的具体的某个学生信息时，会发出相应的SQL去取学生信息  * 这就是典型的N+1问题  * 存在iterator的原因是，有可能会在一个session中查询两次数据，如果使用list每一次都会把所有的对象查询上来  * 而是要iterator仅仅只会查询id，此时所有的对象已经存储在一级缓存(session的缓存)中，可以直接获取  */ Iterator stus = (Iterator)session.createQuery("from Student") .setFirstResult(0).setMaxResults(30).iterate();  for(;stus.hasNext();)  {  Student stu = (Student)stus.next();  System.out.println(stu.getName());  }
```

复制代码

在执行完上述的测试用例后，我们来看看控制台的输出，看会发出多少条 sql 语句：

复制代码

```
Hibernate: select student0_.id as col_0_0_ from t_student student0_ limit ?Hibernate: select student0_.id as id2_0_, student0_.name as name2_0_, student0_.rid as rid2_0_, student0_.sex as sex2_0_ from t_student student0_ where student0_.id=?沈凡Hibernate: select student0_.id as id2_0_, student0_.name as name2_0_, student0_.rid as rid2_0_, student0_.sex as sex2_0_ from t_student student0_ where student0_.id=?王志名Hibernate: select student0_.id as id2_0_, student0_.name as name2_0_, student0_.rid as rid2_0_, student0_.sex as sex2_0_ from t_student student0_ where student0_.id=?叶敦
.........
```

复制代码

我们看到，当如果通过iterator()方法来获得我们对象的时候，hibernate首先会发出1条sql去查询出所有对象的 id 值，当我们如果需要查询到某个对象的具体信息的时候，hibernate此时会根据查询出来的 id 值再发sql语句去从数据

库中查询对象的信息，这就是典型的 N+1 的问题。

那么这种 N+1 问题我们如何解决呢，其实我们只需要使用 list() 方法来获得对象即可。但是既然可以通过 list() 我们就不会出现 N+1的问题，那么我们为什么还要保留 iterator()这种形式呢？我们考虑这样一种情况，如果我们需要在一个session当中要两次查询出很多对象，此时我们如果写两条 list()时，hibernate此时会发出两条 sql 语句，而且这两条语句是一样的，但是我们如果第一条语句使用 list()，而第二条语句使用 iterator()的话，此时我们也会发两条sql语句，但是第二条语句只会将查询出对象的id，所以相对应取出所有的对象而已，显然这样可以节省内存，而如果再要获取对象的时候，因为第一条语句已经将对象都查询出来了，此时会将对象保存到session的一级缓存中去，所以再次查询时，就会首先去缓存中查找，如果找到，则不发sql语句了。这里就牵涉到了接下来这个概念:hibernate的一级缓存。

二、一级缓存(session级别)

我们来看看hibernate提供的一级缓存：

复制代码

```
/**
 * 此时会发出一条sql，将所有学生全部查询出来，并放到session的一级缓存当中
 * 当再次查询学生信息时，会首先去缓存中看是否存在，如果不存在，再去数据库中查询
 * 这就是hibernate的一级缓存(session缓存)
 */
List stus = (List)session.createQuery("from Student")
.setFirstResult(0).setMaxResults(30).list();
Student stu = (Student)session.load(Student.class, 1);
```

复制代码

我们来看看控制台输出：

```
Hibernate: select student0_.id as id2_, student0_.name as name2_, student0_.rid as rid2_, student0_.sex as sex2_ from t_student student0_ limit ?
```

我们看到此时hibernate仅仅只会发出一条 sql 语句，因为第一行代码就会将整个的对象查询出来，放到session的一级缓存中去，当我如果需要再次查询学生对象时，此时首先会去缓存中看是否存在该对象，如果存在，则直接从缓存中取出，就不会再发sql了，但是要注意一点：**hibernate的一级缓存是session级别的，所以如果session关闭后，缓存就没了，此时就会再次发sql去查数据库**。

复制代码

```
try {
    session = HibernateUtil.openSession();
    /**
     * 此时会发出一条sql，将所有学生全部查询出来，并放到session的一级缓存当中
     * 当再次查询学生信息时，会首先去缓存中看是否存在，如果不存在，再去数据库中查询
     * 这就是hibernate的一级缓存(session缓存)
     */
    List stus = (List)session.createQuery("from Student")
    .setFirstResult(0).setMaxResults(30).list();
    Student stu = (Student)session.load(Student.class, 1);
    System.out.println(stu.getName() + "----------");
} catch (Exception e) {
    e.printStackTrace();
} finally {
    HibernateUtil.close(session);
}
/**
 * 当session关闭以后，session的一级缓存也就没有了，这时就又会去数据库中查询
 */
session = HibernateUtil.openSession();
Student stu = (Student)session.load(Student.class, 1);
System.out.println(stu.getName() + "----------");
```

复制代码

复制代码

```
Hibernate: select student0_.id as id2_, student0_.name as name2_, student0_.sex as sex2_, student0_.rid as rid2_ from t_student student0_ limit ?Hibernate: select student0_.id as id2_2_, student0_.name as name2_2_, student0_.sex as sex2_2_, student0_.rid as rid2_2_, classroom1_.id as id1_0_, classroom1_.name as name1_0_, classroom1_.sid as sid1_0_, special2_.id as id0_1_, special2_.name as name0_1_, special2_.type as type0_1_ from t_student student0_ left outer join t_classroom classroom1_ on student0_.rid=classroom1_.id left outer join t_special special2_ on classroom1_.sid=special2_.id where student0_.id=?
```

复制代码

我们看到此时会发出两条sql语句，因为session关闭以后，一级缓存就不存在了，所以如果再查询的时候，就会再发sql。要解决这种问题，我们应该怎么做呢？这就要我们来配置hibernate的二级缓存了，也就是sessionFactory级别的缓存。

三、二级缓存(sessionFactory级别)

使用hibernate二级缓存，我们首先需要对其进行配置，配置步骤如下：

1.hibernate并没有提供相应的二级缓存的组件，所以需要加入额外的二级缓存包，常用的二级缓存包是EHcache。这个我们在下载好的hibernate的lib->optional->ehcache下可以找到(我这里使用的hibernate4.1.7版本)，然后将里面的几个jar包导入即可。

2.在hibernate.cfg.xml配置文件中配置我们二级缓存的一些属性：

```
        <property name="hibernate.cache.use_second_level_cache">trueproperty><property
name="hibernate.cache.region.factory_class">org.hibernate.cache.ehcache.EhCacheRegionFactoryproperty><property
name="hibernate.cache.provider_configuration_file_resource_path">ehcache.xmlproperty>
```

我这里使用的是hibernate4.1.7版本，如果是使用hibernate3的版本的话，那么二级缓存的提供类则要配置成这个：

```
<property name="hibernate.cache.provider_class">net.sf.ehcache.hibernate.EhCacheProviderproperty>
```

3.配置hibernate的二级缓存是通过使用 ehcache的缓存包，所以我们需要创建一个 ehcache.xml 的配置文件，来配置我们的缓存信息，将其放到项目根目录下

复制代码

```
<ehcache>

        <diskStore path="user.dir"/>            <defaultCache        maxElementsInMemory="10000"    //在内存中存放的最大
对象数        eternal="false"                  //是否永久保存缓存，设置成false        timeToIdleSeconds="120"
timeToLiveSeconds="120"                overflowToDisk="true"          //如果对象数量超过内存中最大的数，是否将其保存到磁
盘中，设置成true/>

        <cache name="com.xiaoluo.bean.Student"        maxElementsInMemory="10000"        eternal="false"
timeToIdleSeconds="300"        timeToLiveSeconds="600"        overflowToDisk="true"/><cache name="sampleCache2"
maxElementsInMemory="1000"        eternal="true"        timeToIdleSeconds="0"        timeToLiveSeconds="0"
overflowToDisk="false"/> -->ehcache>
```

复制代码

4.开启我们的二级缓存

①如果使用xml配置，我们需要在 Student.hbm.xml 中加上一下配置：

复制代码

```
package="com.xiaoluo.bean">    <class name="Student" table="t_student">
class="native"/>                                    class>
```

复制代码

二级缓存的使用策略一般有这几种：read-only、nonstrict-read-write、read-write、transactional。**注意**：我们通常使用二级缓存都是将其配置成 read-only ，即我们应当在那些不需要进行修改的实体类上使用二级缓存，否则如果对缓存进行读写的话，性能会变差，这样设置缓存就失去了意义。

②如果使用annotation配置，我们需要在Student这个类上加上这样一个注解：

复制代码

```
@Entity@Table(name="t_student")@Cache(usage=CacheConcurrencyStrategy.READ_ONLY)    //    表示开启二级缓存，并使用read-
only策略publicclass Student{    privateint id;    private String name;    private String sex;    private Classroom room;
.......}
```

复制代码

这样我们的二级缓存配置就算完成了，接下来我们来通过测试用例测试下我们的二级缓存是否起作用

①二级缓存是sessionFactory级别的缓存

TestCase1：

复制代码

```
publicclass TestSecondCache{    @Test    publicvoid testCache1()    {        Session session = null;        try        {
session = HibernateUtil.openSession();            Student stu = (Student) session.load(Student.class, 1);
```

```
System.out.println(stu.getName() + "-----------");        }        catch (Exception e)        {
e.printStackTrace();        }        finally        {            HibernateUtil.close(session);        }        try
{            /**             * 即使当session关闭以后，因为配置了二级缓存，而二级缓存是sessionFactory级别的，所以会从缓存
中取出该数据             * 只会发出一条sql语句             */            session = HibernateUtil.openSession();
Student stu = (Student) session.load(Student.class, 1);            System.out.println(stu.getName() + "-----------");
/**             * 因为设置了二级缓存为read-only，所以不能对其进行修改             */
session.beginTransaction();            stu.setName("aaa");            session.getTransaction().commit();        }
catch (Exception e)        {            e.printStackTrace();            session.getTransaction().rollback();        }
finally        {            HibernateUtil.close(session);        }    }
```

复制代码

```
Hibernate: select student0_.id as id2_2_, student0_.name as name2_2_, student0_.sex as sex2_2_, student0_.rid as
rid2_2_, classroom1_.id as id1_0_, classroom1_.name as name1_0_, classroom1_.sid as sid1_0_, special2_.id as id0_1_,
special2_.name as name0_1_, special2_.type as type0_1_ from t_student student0_ left outer join t_classroom classroom1_
on student0_.rid=classroom1_.id left outer join t_special special2_ on classroom1_.sid=special2_.id where student0_.id=?
aaa-----------aaa-----------
```

因为二级缓存是sessionFactory级别的缓存，我们看到，在配置了二级缓存以后，当我们session关闭以后，我们再去查询对象的时候，此时hibernate首先会去二级缓存中查询是否有该对象，有就不会再发sql了。

②二级缓存缓存的仅仅是对象，如果查询出来的是对象的一些属性，则不会被加到缓存中去

TestCase2：

复制代码

```
    @Test    publicvoid testCache2()    {        Session session = null;        try        {            session =
HibernateUtil.openSession();            /**             * 注意：二级缓存中缓存的仅仅是对象，而下面这里只保存了姓名和
性别两个字段，所以 不会被加载到二级缓存里面             */            List ls = (List) session
.createQuery("select stu.name, stu.sex from Student stu")
.setFirstResult(0).setMaxResults(30).list();        }        catch (Exception e)        {
e.printStackTrace();        }        finally        {            HibernateUtil.close(session);        }        try
{            /**             * 由于二级缓存缓存的是对象，所以此时会发出两条sql             */            session =
HibernateUtil.openSession();            Student stu = (Student) session.load(Student.class, 1);
System.out.println(stu);        }        catch (Exception e)        {            e.printStackTrace();        }    }
```

复制代码

```
Hibernate: select student0_.name as col_0_0_, student0_.sex as col_1_0_ from t_student student0_ limit ?Hibernate:
select student0_.id as id2_2_, student0_.name as name2_2_, student0_.sex as sex2_2_, student0_.rid as rid2_2_,
classroom1_.id as id1_0_, classroom1_.name as name1_0_, classroom1_.sid as sid1_0_, special2_.id as id0_1_,
special2_.name as name0_1_, special2_.type as type0_1_ from t_student student0_ left outer join t_classroom classroom1_
on student0_.rid=classroom1_.id left outer join t_special special2_ on classroom1_.sid=special2_.id where student0_.id=?
```

我们看到这个测试用例，如果我们只是取出对象的一些属性的话，则不会将其保存到二级缓存中去，因为二级缓存缓存的仅仅是对象。

③通过二级缓存来解决 N+1 的问题

TestCase3：

复制代码

```
    @Test    publicvoid testCache3()    {        Session session = null;        try        {            session =
HibernateUtil.openSession();            /**             * 将查询出来的Student对象缓存到二级缓存中去             */
List stus = (List) session.createQuery(                "select stu from Student stu").list();        }        catch
(Exception e)        {            e.printStackTrace();        }        finally        {
HibernateUtil.close(session);        }        try        {            /**             * 由于学生的对象已经缓存在二级缓存
中了，此时再使用iterate来获取对象的时候，首先会通过一条             * 取id的语句，然后在获取对象时去二级缓存中，如果发现
就不会再发SQL，这样也就解决了N+1问题             * 而且内存占用也不多             */            session =
HibernateUtil.openSession();            Iterator iterator = session.createQuery("from Student")
.iterate();            for (; iterator.hasNext();)            {                Student stu = (Student) iterator.next();
System.out.println(stu.getName());            }        }        catch (Exception e)        {
e.printStackTrace();        }    }
```

复制代码

当我们如果需要查询出两次对象的时候，可以使用二级缓存来解决N+1的问题。

④二级缓存会缓存 hql 语句吗？

TestCase4：

复制代码

```
    @Test    publicvoid testCache4()    {        Session session = null;        try        {            session = HibernateUtil.openSession();            List ls = session.createQuery("from Student")
.setFirstResult(0).setMaxResults(50).list();        }        catch (Exception e)        {
e.printStackTrace();        }        finally        {            HibernateUtil.close(session);        }        try        {            /**             * 使用List会发出两条一模一样的sql，此时如果希望不发sql就需要使用查询缓存             */
session = HibernateUtil.openSession();            List ls = session.createQuery("from Student")
.setFirstResult(0).setMaxResults(50).list();            Iterator stu = ls.iterator();            for(;stu.hasNext();)
{                Student student = stu.next();                System.out.println(student.getName());            }
}        catch (Exception e)        {            e.printStackTrace();        }        finally        {
HibernateUtil.close(session);        }    }
```

复制代码

```
Hibernate: select student0_.id as id2_, student0_.name as name2_, student0_.sex as sex2_, student0_.rid as rid2_ from t_student student0_ limit ?Hibernate: select student0_.id as id2_, student0_.name as name2_, student0_.sex as sex2_, student0_.rid as rid2_ from t_student student0_ limit ?
```

我们看到，当我们如果通过 list() 去查询两次对象时，二级缓存虽然会缓存查询出来的对象，但是我们看到发出了两条相同的查询语句，这是因为二级缓存不会缓存我们的hql查询语句，要想解决这个问题，我们就要配置我们的查询缓存了。

四、查询缓存(sessionFactory级别)

我们如果要配置查询缓存，只需要在hibernate.cfg.xml中加入一条配置即可：

```
        <property name="hibernate.cache.use_query_cache">trueproperty>
```

然后我们如果在查询hql语句时要使用查询缓存，就需要在查询语句后面设置这样一个方法:

```
List ls = session.createQuery("from Student where name like ?")                    .setCacheable(true)    //开启查询缓存，查询缓存也是SessionFactory级别的缓存                    .setParameter(0, "%王%")
.setFirstResult(0).setMaxResults(50).list();
```

如果是在annotation中，我们还需要在这个类上加上这样一个注解：@Cacheable

接下来我们来通过测试用例来看看我们的查询缓存

①查询缓存也是sessionFactory级别的缓存

TestCase1：

复制代码

```
    @Test    publicvoid test2() {        Session session = null;        try {            /**             * 此时会发出一条sql取出所有的学生信息             */            session = HibernateUtil.openSession();            List ls =
session.createQuery("from Student")                    .setCacheable(true)  //开启查询缓存,查询缓存也是sessionFactory级别的缓存                    .setFirstResult(0).setMaxResults(50).list();            Iterator stus = ls.iterator();
for(;stus.hasNext();) {                Student stu = stus.next();                System.out.println(stu.getName());
}        } catch (Exception e) {            e.printStackTrace();        } finally {
HibernateUtil.close(session);        }        try {            /**             * 此时会发出一条sql取出所有的学生信息
*/            session = HibernateUtil.openSession();            List ls = session.createQuery("from Student")
.setCacheable(true)  //开启查询缓存,查询缓存也是sessionFactory级别的缓存
.setFirstResult(0).setMaxResults(50).list();            Iterator stus = ls.iterator();            for(;stus.hasNext();)
{                Student stu = stus.next();                System.out.println(stu.getName());            }        }
catch (Exception e) {            e.printStackTrace();        } finally {            HibernateUtil.close(session);
}    }
```

复制代码

```
Hibernate: select student0_.id as id2_, student0_.name as name2_, student0_.sex as sex2_, student0_.rid as rid2_ from t_student student0_ limit ?
```

我们看到，此时如果我们发出两条相同的语句，hibernate也只会发出一条sql，因为已经开启了查询缓存了，并且查询

缓存也是sessionFactory级别的

②只有当 HQL 查询语句完全相同时，连参数设置都要相同，此时查询缓存才有效

TestCase2：

复制代码

```
    @Test    publicvoid test3() {        Session session = null;        try {            /**             * 此时会发出一条sql取出所有的学生信息             */            session = HibernateUtil.openSession();            List ls = session.createQuery("from Student where name like ?")                    .setCacheable(true)//开启查询缓存，查询缓存也是SessionFactory级别的缓存                    .setParameter(0, "%王%")
.setFirstResult(0).setMaxResults(50).list();            Iterator stus = ls.iterator();            for(;stus.hasNext();)
{                Student stu = stus.next();                System.out.println(stu.getName());            }        }
catch (Exception e) {            e.printStackTrace();        } finally {            HibernateUtil.close(session);
}        session = null;        try {            /**             * 此时会发出一条sql取出所有的学生信息             */
session = HibernateUtil.openSession();            /**             * 只有当HQL完全相同的时候，连参数都要相同，查询缓存才有效             *///            List ls = session.createQuery("from Student where name like ?")//
.setCacheable(true)//开启查询缓存，查询缓存也是SessionFactory级别的缓存//                    .setParameter(0, "%王%")//
.setFirstResult(0).setMaxResults(50).list();            List ls = session.createQuery("from Student where name like ?")
.setCacheable(true)//开启查询缓存，查询缓存也是SessionFactory级别的缓存                    .setParameter(0, "%张%")
.setFirstResult(0).setMaxResults(50).list();            Iterator stus = ls.iterator();            for(;stus.hasNext();)
{                Student stu = stus.next();                System.out.println(stu.getName());            }        }
catch (Exception e) {            e.printStackTrace();        } finally {            HibernateUtil.close(session);
}    }
```

复制代码

```
Hibernate: select student0_.id as id2_, student0_.name as name2_, student0_.sex as sex2_, student0_.rid as rid2_ from t_student student0_ where student0_.name like ? limit ?Hibernate: select student0_.id as id2_, student0_.name as name2_, student0_.sex as sex2_, student0_.rid as rid2_ from t_student student0_ where student0_.name like ? limit ?
```

我们看到，如果我们的hql查询语句不同的话，我们的查询缓存也没有作用

③查询缓存也能引起 N+1 的问题

查询缓存也能引起 N+1 的问题，我们这里首先先将 Student 对象上的二级缓存先注释掉：

TestCase4：

复制代码

```
    @Test    publicvoid test4() {        Session session = null;        try {            /**             * 查询缓存缓存的不是对象而是id             */            session = HibernateUtil.openSession();            List ls = session.createQuery("from Student where name like ?")                    .setCacheable(true)//开启查询缓存，查询缓存也是SessionFactory级别的缓存                    .setParameter(0, "%王%")
.setFirstResult(0).setMaxResults(50).list();            Iterator stus = ls.iterator();            for(;stus.hasNext();)
{                Student stu = stus.next();                System.out.println(stu.getName());            }        }
catch (Exception e) {            e.printStackTrace();        } finally {            HibernateUtil.close(session);
}                session = null;        try {            /**             * 查询缓存缓存的是id，此时由于在缓存中已经存在了这样的一组学生数据，但是仅仅只是缓存了             * id，所以此处会发出大量的sql语句根据id取对象，这也是发现N+1问题的第二个原因             * 所以如果使用查询缓存必须开启二级缓存             */            session = HibernateUtil.openSession();            List ls = session.createQuery("from Student where name like ?")
.setCacheable(true)//开启查询缓存，查询缓存也是SessionFactory级别的缓存                    .setParameter(0, "%王%")
.setFirstResult(0).setMaxResults(50).list();            Iterator stus = ls.iterator();            for(;stus.hasNext();)
{                Student stu = stus.next();                System.out.println(stu.getName());            }        }
catch (Exception e) {            e.printStackTrace();        } finally {            HibernateUtil.close(session);
}    }
```

复制代码

复制代码

```
Hibernate: select student0_.id as id2_, student0_.name as name2_, student0_.sex as sex2_, student0_.rid as rid2_ from t_student student0_ where student0_.name like ? limit ?Hibernate: select student0_.id as id2_2_, student0_.name as
```

```
name2_2_, student0_.sex as sex2_2_, student0_.rid as rid2_2_, classroom1_.id as id1_0_, classroom1_.name as name1_0_,
classroom1_.sid as sid1_0_, special2_.id as id0_1_, special2_.name as name0_1_, special2_.type as type0_1_ from
t_student student0_ left outer join t_classroom classroom1_ on student0_.rid=classroom1_.id left outer join t_special
special2_ on classroom1_.sid=special2_.id where student0_.id=?Hibernate: select student0_.id as id2_2_, student0_.name
as name2_2_, student0_.sex as sex2_2_, student0_.rid as rid2_2_, classroom1_.id as id1_0_, classroom1_.name as name1_0_,
classroom1_.sid as sid1_0_, special2_.id as id0_1_, special2_.name as name0_1_, special2_.type as type0_1_ from
t_student student0_ left outer join t_classroom classroom1_ on student0_.rid=classroom1_.id left outer join t_special
special2_ on classroom1_.sid=special2_.id where student0_.id=?Hibernate: select student0_.id as id2_2_, student0_.name
as name2_2_, student0_.sex as sex2_2_, student0_.rid as rid2_2_, classroom1_.id as id1_0_, classroom1_.name as name1_0_,
classroom1_.sid as sid1_0_, special2_.id as id0_1_, special2_.name as name0_1_, special2_.type as type0_1_ from
t_student student0_ left outer join t_classroom classroom1_ on student0_.rid=classroom1_.id left outer join t_special
special2_ on classroom1_.sid=special2_.id where student0_.id=?Hibernate: select student0_.id as id2_2_, student0_.name
as name2_2_, student0_.sex as sex2_2_, student0_.rid as rid2_2_, classroom1_.id as id1_0_, classroom1_.name as name1_0_,
classroom1_.sid as sid1_0_, special2_.id as id0_1_, special2_.name as name0_1_, special2_.type as type0_1_ from
t_student student0_ left outer join t_classroom classroom1_ on student0_.rid=classroom1_.id left outer join t_special
special2_ on classroom1_.sid=special2_.id where student0_.id=?........................
```

复制代码

我们看到，当我们将二级缓存注释掉以后，在使用查询缓存时，也会出现 N+1 的问题，为什么呢？

因为**查询缓存缓存的也仅仅是对象的id**，所以第一条 sql 也是将对象的id都查询出来，但是当我们后面如果要得到每个对象的信息的时候，此时又会发sql语句去查询，所以，如果要使用查询缓存，我们一定也要开启我们的二级缓存，这样就不会出现 N+1 问题了

好了，整篇随笔大概花费了2个小时来编写，可以说将hibernate的 N+1 问题、一级缓存、二级缓存、查询缓存的概念以及可能出现的问题都分析了透，希望能对大家提供帮助！