Hibernate03

1. 二级缓存

公开的, 基本不变, 不太紧要的数据

1.1 简介

> *二级缓存称 SessionFactory 级的缓存，二级缓存可以被所有的session共享。*
>
> *二级缓存的生命周期和 SessionFactory 的一致。*
>
> *本身hibernate也提供二级缓存的工具，但是一般我们在实际开发中不使用，而是使用一些外部的二级缓存工具.*
>
> *二级缓存的位置：SessionFactoryImpl/ 也是 map, 也有集合缓存*

1.2 配置 Hibernate_03_1_SessionFactoryCache

1) 利用 EhCahe 实现二级缓存

- 1.首先当然是加入相关的jar包，同时将配置文件ehcache.xml加入到src下。
- 2.开启二级缓存，修改hibernate.cfg.xml文件
- 3.指定缓存产品提供商
- 4.指定哪些实体使用二级缓存

```
<!-- 开启二级缓存 -->
<property name="hibernate.cache.use_second_level_cache">true</property>


<!-- 指定缓存产品提供商 -->
Hibernate 4.0以上设置 factory
<property name="cache.region.factory_class">org.hibernate.cache.ehcache
```

```
.EhCacheRegionFactory</property>


<!-- 指定哪些实体使用二级缓存 -->
<class-cache class="com.lanou.domain.Clazz" usage="read-only"/>
<!-- usage 属性是必须的指定并发访问策略，取值为 transactional(事务缓存)、read-w
rite(读写缓存)、nonstrict-read-write(非严格读写缓存)、read-only(只读缓存)。--
>
```

read-only这种策略表示只读，但是这样就不能更改缓存，但是有时候我们修改了数据库数据而缓存不能更改
确实会造成错误，这里我们可以在ehcache.xml中对缓存的生命周期进行配置，一定时间后让缓存更新。

1.3 操作

哪些方法可以把对象放在二级缓存中?
[1] get

```
/* session.get 不仅可以把数据放入一级缓存中, 还可以放入二级缓存中.
   先从一级缓存中查找, 然后找二级缓存, 找不到再去数据库查找 */
@Test
public void testGet(){
    Session session = HibernateUtil.openSession();
    Clazz clazz = session.get(Clazz.class, 1L);
    session.close();
    session = HibernateUtil.openSession();
    Clazz clazzz = session.get(Clazz.class, 1L);
    session.close();
}
```

[2] save

```
/*    session.save 不操作二级缓存! */
@Test
public void testSave(){
    Session session = sessionFactory.openSession();
    Transaction transaction = session.beginTransaction();
    Clazz clazz = new Clazz("222","222222");
    session.save(clazz);
    // 在这里使用 二级缓存的统计机制, 默认没有打开..
    // 输出 0 !
    System.out.println(sessionFactory.getStatistics().getEntityLoadCoun
```

```
t());
    transaction.commit();
    session.close();
}
```

```
<!-- 二级缓存统计机制-->
<property name="generate_statistics">true</property>
```

[3] query

```
// 利用查询让对象进入二级缓存中, 但是不能够取数据
@Test
public void testQuery(){
    Session session = sessionFactory.openSession();
    // 需要放置 HQL 语句 // Hibernate Query Language . 使用全类名
    List<Clazz> list = session.createQuery("from com.lanou.domain.Clazz
").list();
    System.out.println(sessionFactory.getStatistics().getEntityLoadCoun
t());
    session.close();

    session = sessionFactory.openSession();
     list = session.createQuery("from com.lanou.domain.Clazz").list();
    System.out.println(sessionFactory.getStatistics().getEntityLoadCoun
t());
    session.close();
}
```

```
// 查询
@Test
public void testQuery(){
    Session session = sessionFactory.openSession();
    // 需要放置 HQL 语句 // Hibernate Query Language . 使用全类名
    /* list 方法可以让 HQL 语句指定的对象进入二级缓存中. 但是 list 方法不利用二级
缓存查询数据     */
     List<Clazz> list = session.createQuery("from com.lanou.domain.Claz
z").list();
    System.out.println(sessionFactory.getStatistics().getEntityLoadCoun
t());
    session.close();

    /* iterate 先查找该表中所有 id 值.
       再利用id 值从二级缓存中查找数据, 如果有则利用二级缓存.
       如果没有则根据 id 查询该表中数据 */
      session = sessionFactory.openSession();
```

```
     Iterator<Clazz> iterate = session.createQuery("from com.lanou.doma
in.Clazz").iterate();
     while (iterate.hasNext()){
         System.out.println(iterate.next().getCname());
     }
    session.close();
}
```

集合放入二级缓存中

```
<set name="students" cascade="all" inverse="false">
        <cache usage="read-only"/>
```

```
// 集合的二级缓存
@Test
public void  collectionTest(){
      Session session =  sessionFactory.openSession();
      Clazz clazz = session.get(Clazz.class, 1L);
      Set<Student> students = clazz.getStudents();
    for (Student stu: students){
        System.out.println(stu.getSname());
    }
// 输出值为1, 有一个集合进入二级缓存中    System.out.println(sessionFactory.g
etStatistics().getCollectionLoadCount());
  session.close();
}
```

哪些方法可以把对象放入: get , list , iterate
哪些方法可以把对象取出: get , iterate

1.4 二级缓存 - 缓存到磁盘上

缓存的内容过多, 查询缓存的速度越来越慢
更改 ehcache.xml

```
   maxElementsInMemory  - 设置缓存最大数目
      eternal              - 设置缓存中的数据是否失效，true为永不过期。
      timeToIdleSeconds    - 指定缓存多久未被使用便清理掉
      timeToLiveSeconds    - 指定缓存的生命长度，单位是秒
      overflowToDisk       - 是否保存到磁盘，当系统当机时
<diskStore path="/Users/dllo/Desktop/Temp"/>
<cache
    name="com.lanou.domain.Clazz”
    maxElementsInMemory="2”
    eternal="false”
    timeToIdleSeconds="120”
    timeToLiveSeconds="120”
    overflowToDisk="true"/>
```

2. 查询缓存 QueryCache

2.1 概述

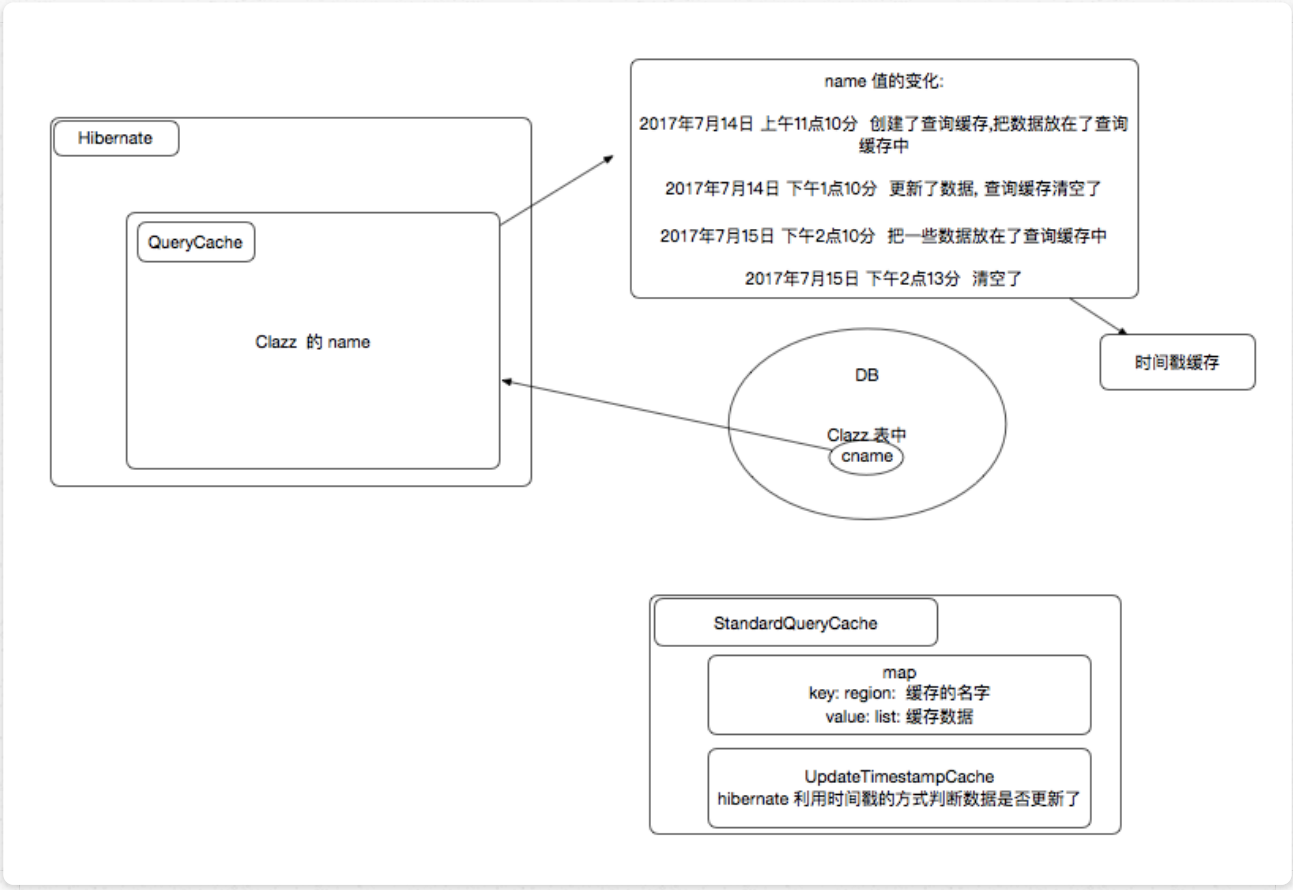
一级缓存, 二级缓存都是对象缓存. 会把数据库中所有的字段都查询出来.
如果字段很多, 而程序中使用的又很少, 效率就会很低.
查询缓存: 也叫做数据缓存. 内存中需要多少数据就把多少数据放入查询缓存中

2.2 生命周期

只要一些数据放入到查询缓存中, 该缓存一直存在, 直到查询缓存中的数据被修改了, 该缓存的生命周期结束

2.3 内部实现

在 sessionFactoryImpl 中有 QueryCache, UpdateTimestampsCache
查询缓存中使用时间戳缓存判断数据是否更新



2.4 查询缓存的使用

- 在hibernate.cfg.xml文件中启用查询缓存，如：
- <property name="hibernate.cache.use_query_cache">true</property>
- 因为hibernate中查询缓存默认是关闭的，这和二级缓存不一样。
- 在程序中必须手动启用查询缓存，如：
- query.setCacheable(true);

[1] list, 查询对象

```
/*    "from clazz" 可以已放入到查询缓存中 */
@Test
public void queryTest(){
    Session session = sessionFactory.openSession();
    Query query = session.createQuery("from com.lanou.domain.Clazz");
    query.setCacheable(true);// 说明 query 要使用查询缓存
    query.list(); // 把数据放入了查询缓存中    session.close();

    session = sessionFactory.openSession();
    query = session.createQuery("from com.lanou.domain.Clazz");
    query.setCacheable(true);
    query.list();
```

```
    session.close();
}
```

[2] list, 查询某列数据

```
/*"select cname from Clazz”
    查询出来的放到查询缓存中, 但是进不到二级缓存中, 因为不是对象 */
@Test
public void queryTest2(){
    Session session = sessionFactory.openSession();
    Query query = session.createQuery("select cname from Clazz");
    query.setCacheable(true);// 说明 query 要使用查询缓存
    query.list(); // 把数据放入了查询缓存中
    System.out.println(sessionFactory.getStatistics().getEntityLoadCoun
t());
    session.close();

    session = sessionFactory.openSession();
    query = session.createQuery("select cname from Clazz");
    query.setCacheable(true);
    query.list();
    session.close();
}
```

[3] list: HQL 语句不同

```
/*    如果两个 hql 语句一样, 才可以利用查询缓存. 有一点不一样就不能利用 */
@Test
public void queryTest3(){
    Session session = sessionFactory.openSession();
    Query query = session.createQuery("select cname from Clazz");
    query.setCacheable(true);// 说明 query 要使用查询缓存
    query.list(); // 把数据放入了查询缓存中
    System.out.println(sessionFactory.getStatistics().getEntityLoadCoun
t());
    session.close();

    session = sessionFactory.openSession();
    query = session.createQuery("select cname from Clazz where cid=1");

    query.setCacheable(true);
    query.list();
    session.close();
}
```

[4] 验证生命周期

```
/*    先把一些数据放入查询缓存中, 然后修改, 查看生命周期 */
@Test
public void queryTest4(){
    // 把 name 放入到查询缓存中
    Session session = sessionFactory.openSession();
    Query query = session.createQuery("select cname from Clazz");
    query.setCacheable(true);// 说明 query 要使用查询缓存
    query.list(); // 把数据放入了查询缓存中
    session.close();

    // 修改
    // 修改了查询缓存的时间戳缓存,从而知道了已经被修改, 然后给清空了
    session = sessionFactory.openSession();
    Transaction transaction = session.beginTransaction();
    Clazz clazz = session.get(Clazz.class, 1L);
    clazz.setCname("11");
    transaction.commit();
    session.close();

    // 再次查询. 因为已经修改, 所以 name 都被清空了
    session = sessionFactory.openSession();
    query = session.createQuery("select cname from Clazz");
    query.setCacheable(true);
    query.list();
    session.close();
}
```

总结

一级缓存: 再一次请求中尽量减少和数据库的交互次数. 在 session.flush 之前, 改变的是一级缓存的对象属性. 当 session.flush 时才和数据库交互, 一级缓存解决不了重复查询问题. 一级缓存是对象缓存.

二级缓存: 把 经常不改变, 常用的, 公共的数据放入进来. 可以重复查询. 利用 get/iterate 可以得到二级缓存数据

查询缓存: 可以缓存对象或数据. 可以利用 list 可以添加得到查询缓存数据. 查询缓存中放的是数据, 是数据缓存.