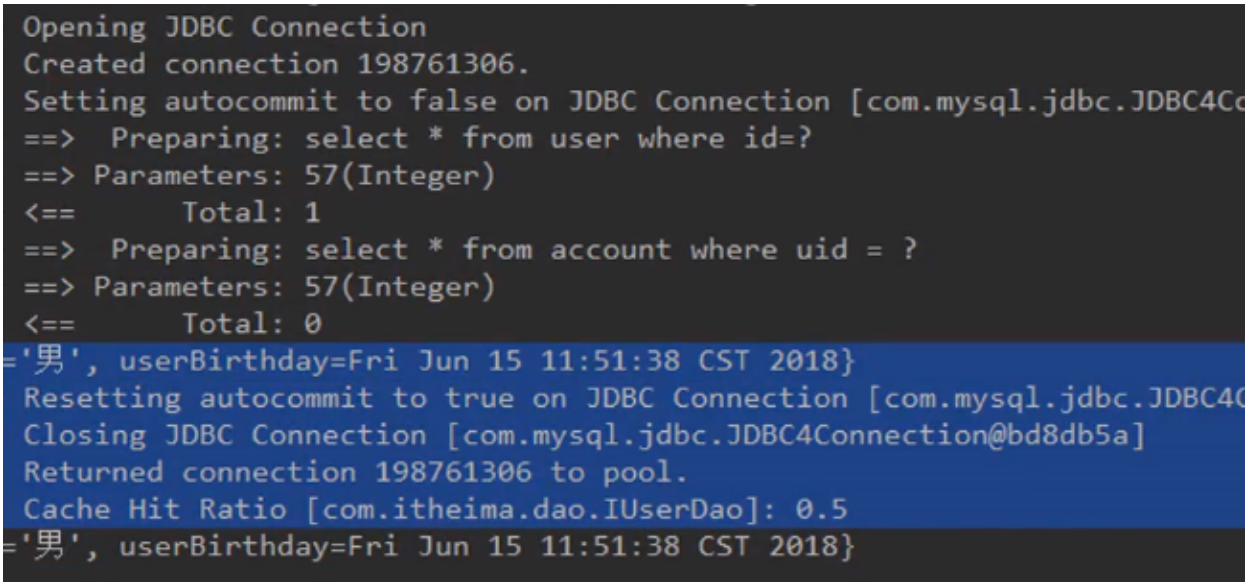
缓冲

一、缓冲概念

即开启缓冲后，第一次查询完，第二次查询直接使用缓冲的数据，不用再建立连接



1、缓冲：存在于内存中的临时数据

2、为什么使用缓冲

查询缓存来缓存数据，从而达到提高查询性能的要求，以提高我们项目的效率！

3、什么样的数据能使用缓存，什么样的数据不能使用

适用于缓存：

经常查询并且不经常改变的。

数据的正确与否对最终结果影响不大的。

不适用于缓存：

经常改变的数据

数据的正确与否对最终结果影响很大的。

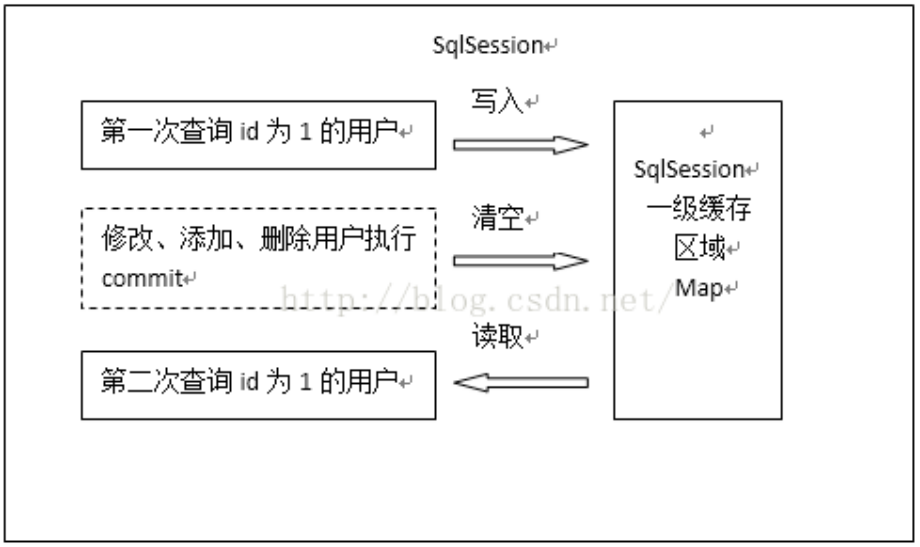
4、mybatis的缓存机制有两级：

（1）一级缓存：一级缓存mybatsi已近为我们自动开启，不用我们手动操作，而且我们是关闭不了的！但是我们可以手动清除缓存。（SqlSession级别）

（2）二级缓存：二级缓存需要我们手动开启。（全局级别）

二、一级缓冲

1、图解



2、一级缓存的生命周期

a、MyBatis在开启一个数据库会话时，会创建一个新的SqlSession对象，SqlSession对象中会有一个新的Executor对象。Executor对象中持有一个新的PerpetualCache对象；当会话结束时，SlSession对象及其内部的Executor对象还有PerpetualCache对象也一并释放掉。

b、如果SqlSession调用了close()方法，会释放掉一级缓存PerpetualCache对象，一级缓存将不可用。

```
@Test
 public void testFirstLevelCache(){
     User user1 = userDao.findById(41);
     System.out.println(user1);

     sqlSession.close();
     //再次获取SqlSession对象
    sqlSession = factory.openSession();
    userDao = sqlSession.getMapper(IUserDao.class);

     User user2 = userDao.findById(41);
     System.out.println(user2);
     System.out.println(user1 == user2);
```

```
}
false
```

c、如果SqlSession调用了clearCache()，会清空PerpetualCache对象中的数据，但是该对象user1仍可使用。

```
@Test
public void testFirstLevelCache(){
    User user1 = userDao.findById(41);
    System.out.println(user1);

    sqlSession.clearCache();//此方法也可以清空缓存

    userDao = sqlSession.getMapper(IUserDao.class);
    User user2 = userDao.findById(41);
    System.out.println(user2);
    System.out.println(user1 == user2);
}
//false
```

d、SqlSession中执行了任何一个update操作(update()、delete()、insert())，都会清空PerpetualCache对象的数据，但是该对象可以继续使用

```
@Test
public void testClearlCache(){
    //1.根据id查询用户
    User user1 = userDao.findById(41);
    System.out.println(user1);

    //2.更新用户信息
    user1.setUsername("update user clear cache");
    user1.setAddress("北京市海淀区");
    userDao.updateUser(user1);

    //3.再次查询id为41的用户
    User user2 = userDao.findById(41);
    System.out.println(user2);

    System.out.println(user1 == user2);
}
```

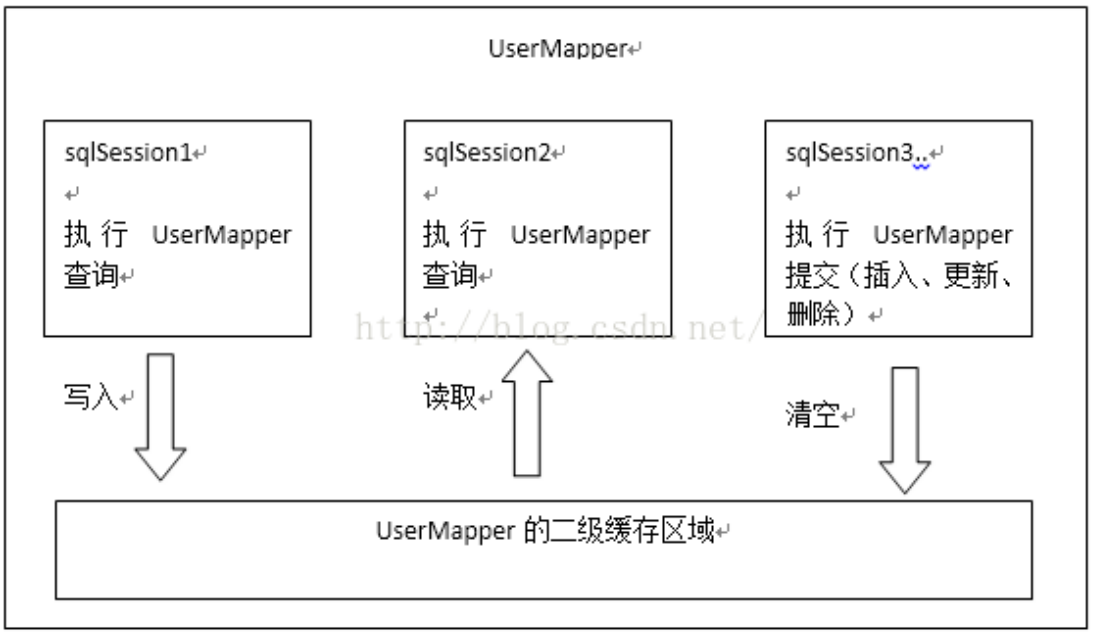
三、二级缓冲

1、概念

它指的是Mybatis中SqlSessionFactory对象的缓存。由同一个SqlSessionFactory对象创建的SqlSession共享其缓存。

注意：实现二级缓存的时候，MyBatis要求返回的POJO必须是可序列化的，也就是要求实现Serializable接口

2、图解



3、二级缓存的使用步骤：

第一步：让Mybatis框架支持二级缓存（在SqlMapConfig.xml中配置）

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
      PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
      "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
   <properties resource="jdbcConfig.properties"></properties>

  //配置支持二级缓存
   <settings>
      <setting name="cacheEnabled" value="true"/>
   </settings>

    <!--使用typeAliases配置别名，它只能配置domain中类的别名 -->
   <typeAliases>
      <package name="com.itheima.domain"></package>
   </typeAliases>
</configuration>
```

第二步：让当前的映射文件支持二级缓存（在IUserDao.xml中配置）

第三步：让当前的操作支持二级缓存（在select标签中配置）

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.itheima.dao.IUserDao">

    <!--开启user支持二级缓存-->
    <cache/>

    <!-- 根据id查询用户 -->
    <select id="findById" parameterType="INT" resultType="user"
useCache="true">
        select * from user where id = #{uid}
    </select>
</mapper>
```

4、如果我们配置了二级缓存就意味着：

- 映射语句文件中的所有select语句将会被缓存。
- 映射语句文件中的所欲insert、update和delete语句会刷新缓存。
- 缓存会使用默认的Least Recently Used（LRU，最近最少使用的）算法来收回。
- 根据时间表，比如No Flush Interval,（CNFI没有刷新间隔），缓存不会以任何时间顺序来刷新。
- 缓存会存储列表集合或对象(无论查询方法返回什么)的1024个引用
- 缓存会被视为是read/write(可读/可写)的缓存，意味着对象检索不是共享的，而且可以安全的被调用者修改，不干扰其他调用者或线程所做的潜在修改。