# 事务以及hibernate中的缓存

1. 事务的概念：

一组必须要同时完成的或者同时取消的操作组成

事务就是一个事情有开始和结束，在这期间所做的事情，只要没结束都是可以回来的。即：事务回滚。数据库中，对数据库中数据的操作，肯定有可能会出错，所以需要事务回滚这个机制。还有，多个用户对数据进行操作的时候，有没有可能会发生错误呢？是很有可能的，比如，两个人都同时读取到了一条数据，一个改变了，其中一项，而另一个人改变了另一项，那么最后两个人提交的时候，就会发生后一个的改变覆盖前一个（丢失更新）。所以，通过设置事务隔离的级别，来避免这种现象。

1. 事务的隔离级别:

事务具有4个特性：一致性，原子性，隔离性，持久性。

隔离性：**一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。**

为了防止事务的并发问题，数据库对于事务的隔离级别设置了4种：

**- 读取未提交(Read uncommitted)：处于此模式下可能会出现脏读、幻象读、不可重复读**

**- 读取已提交(Read committed)：处于此模式下可能会出现幻象读、不可重复读**

**- 可重复读(Repeatable read)：处于此模式下可能会出现幻象读**

**- 串行(Serialize)：不会出现幻象读**

三．事务中可能发生的问题

3.1存在的问题

脏读：

**一个事务读取到了另外一个事务没有提交的数据  
            事务1：更新一条数据  
                             ------------->事务2：读取事务1更新的记录  
            事务1：调用commit进行提交  
            \*\*\*此时事务2读取到的数据是保存在数据库内存中的数据，称为脏读。  
            \*\*\*读到的数据为脏数据**

不可重复读：

**在同一事务中，两次读取同一数据，得到内容不同  
            事务1：查询一条记录  
                            -------------->事务2：更新事务1查询的记录  
                            -------------->事务2：调用commit进行提交  
            事务1：再次查询上次的记录  
            \*\*\*此时事务1对同一数据查询了两次，可得到的内容不同，称为不可重复读**

幻读：

**同一事务中，用同样的操作读取两次，得到的记录数不相同  
            事务1：查询表中所有记录  
                              -------------->事务2：插入一条记录  
                              -------------->事务2：调用commit进行提交  
            事务1：再次查询表中所有记录**

丢失更新：

* 上面几种都是发生在读取数据的时候，而丢失更新是发生在写数据的时候，

**两个人都同时读取到了一条数据，一个改变了，其中一项，而另一个人改变了另一项，那么最后两个人提交的时候，就会发生后一个的改变覆盖前一个（丢失更新）**

3.2解决办法：

对于读问题：解决办法就是设置事务隔离级别。

1. **read uncommitted**
2. **read committed**
3. **repeatable read**
4. **Serialize**

而对于写问题：

1. 悲观锁：即假设所有的事务都会丢失更新

Select \* from user **for update 这就是数据库自带的锁机制，悲观锁。**

**大并发的情况下，不推荐使用悲观锁，因为太低效率了。**

1. 乐观锁：不采用数据库的锁机制，而是自己设置一个版本号字段，读取一次版本号变成一次。

四．Hibernate中的缓存

缓存：通常是用来程序和物理存储数据源之间的一种中介。以减少程序对物理源的访问，提高程序效率。

缓存的介质一般是内存，所以读写速度很快。但如果缓存中存放的数据量非常大时，也会用硬盘作为缓存介质。缓存的实现不仅仅要考虑存储的介质，还要考虑到管理缓存的并发访问和缓存数据的生命周期。

4.1缓存的范围：

1.事务范围：**缓存的生命周期就是一个事务的存在范围。（一级缓存）**

2.进程范围：**缓存的数据一般被一个进程内所有事务共享。这些事务有可能是并发 访问缓存，因此必须对缓存采取必要的事务隔离机制。（二级缓存）**

3.集群范围：**缓存的数据一般在多个机器中共享。缓存中的数据被复制到集群环境 中的每个进程节点，进程间通过远程通信来保证缓存中的数据的一致 性，缓存中的数据通常采用对象的松散数据形式。**

所以Hibernate中二级缓存需要设置 事务隔离机制。

4.2缓存的一般实现方式：

简单来说，就是把数据复制多份存在集合中。当需要用的时候，从集合中取出。当然缓存还要考虑生命周期，考虑并发事务隔离机制，等等。

4.3 Hibernate的三种缓存：

**1.一级缓存：hibernate自带的缓存，属于事务型缓存，不需要配置。**

（通过一级缓存，可以实现自动更新功能）

当向数据库中获取数据的时候，会把获取的数据存在一级缓存区中的缓存区以及快照区。

快照区主要用来对比缓存区，用来实现 自动更新用的

（**一旦缓存区改变后，与快照区中的数据对比，如果不相同，那么就会向数据库中更新 数据，同时如果存在二级缓存就 会自动把更新后的数据同步到二级缓存区域）**

Customer customer=(Customer)session.get(Customer.class,1); 获取数据同时存入一级 缓存区域。

一级缓存，不能取消，他会随着Session 生命周期结束而结束，网上都是这么说的！！

但是，这样说是不准确的，实际上,一级缓存是一个 事务范围的缓存 因此，实际情况是 一级缓存的生命周期是随着一个session的事务结束而结束。

因为，绑定线程的session中，不同事务中，一级缓存不能共用。

**2.二级缓存：需要配置的缓存，可以不使用。通过配置文件开启。**

<!--二级缓存的使用配置-->  
<**property** name="hibernate.cache.provider\_class">  
 org.hibernate.cache.EhCacheProvider</**property**>  
<!--二级缓存的开启-->  
<**property** name="hibernate.cache.use\_second\_level\_cache">true</**property**>

哪些类要使用二级缓存：

<**class-cache** class="bean.Customer" usage="read-write"/>  
<**class-cache** class="bean.Order" usage="read-write" />  
<!--哪些集合要使用二级缓存-->  
<**collection-cache** collection="bean.Customer.orders" usage="read-write"/>

二级缓存属于进程范围的缓存，因此可以存储在 内存 或者 硬盘中，同时，数据存储的形式是 实体的松散数据。 只能通过，session.get()方法来使用，因为它主要是用id来使用，当使用session.createQuery().list(),方法，可以往二级缓存中存储数据，但是再使用list()方法还是需要发送sql从数据库中获取。