<http://www.blogjava.net/fhtdy2004/archive/2009/08/01/292542.html>

**设置隔离级别**

设置一个事务的隔离级别

* SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
* SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
* SET TRANSACTION READ ONLY;

设置增个会话的隔离级别

* ATER SESSION SET ISOLATION\_LEVE SERIALIZABLE;
* ATER SESSION SET ISOLATION\_LEVE READ COMMITTED;

oracle中锁的类型：  
 然后再读时不需要加锁，这一点Oracle的共享锁的实现与上一篇中的共享锁原理有点不同，  
  
今天在查看oracle官方文档lock的那一部分的时候发现一个新的概念，isolation level (数据隔离级别)，虽然以前学过，但是忘的差不多了。  
隔离级别（isoation eve）

隔离级别定义了事务与事务之间的隔离程度。

隔离级别与并发性是互为矛盾的：隔离程度越高，数据库的并发性越差；隔离程度越低，数据库的并发性越好。

ANSI/ISO SQ92标准定义了一些数据库操作的隔离级别：

* 未提交读（read uncommitted）
* 提交读（read committed）
* 重复读（repeatabe read）
* 序列化（seriaizabe）

通过一些现象，可以反映出隔离级别的效果。这些现象有：

* 更新丢失（ost update）：当系统允许两个事务同时更新同一数据是，发生更新丢失。
* 脏读（dirty read）：当一个事务读取另一个事务尚未提交的修改时，产生脏读。
* 非 重复读（nonrepeatabe read）：同一查询在同一事务中多次进行，由于其他提交事务所做的修改或删除，每次返回不同的结果集，此时发生非重复读。(A transaction rereads data it has previousy read and finds that another committed transaction has modified or deeted the data. )
* 幻 像（phantom read）：同一查询在同一事务中多次进行，由于其他提交事务所做的插入操作，每次返回不同的结果集，此时发生幻像读。(A transaction reexecutes a query returning a set of rows that satisfies a search condition and finds that another committed transaction has inserted additiona rows that satisfy the condition. )

下面是隔离级别及其对应的可能出现或不可能出现的现象

|  |  |  |  |
| --- | --- | --- | --- |
|  | Dirty Read | NonRepeatabe Read | Phantom Read |
| Read uncommitted | Possible | Possible | Possible |
| Read committed | not possible | Possible | Possible |
| Repeatabe read | not possible | not possible | Possible |
| Seriaizabe | not possible | not possible | not possible |

**ORACE的隔离级别**

ORACE提供了SQ92标准中的read committed和seriaizabe，同时提供了非SQ92标准的read-ony。

**read committed：**

* 这是ORACE缺省的事务隔离级别。
* 事务中的每一条语句都遵从语句级的读一致性。
* 保证不会脏读；但可能出现非重复读和幻像。

**seriaizabe：（**串行执行事务，并发性最小）

* 简单地说，seriaizabe就是使事务看起来象是一个接着一个地顺序地执行。
* 仅仅能看见在本事务开始前由其它事务提交的更改和在本事务中所做的更改。
* 保证不会出现非重复读和幻像。
* Seriaizabe隔离级别提供了read-ony事务所提供的读一致性（事务级的读一致性），同时又允许DM操作。

如果有在seriaizabe事务开始时未提交的事务在seriaizabe事务结束之前修改了seriaizabe事务将要修改的行并进行了提交，则seriaizabe事务不会读到这些变更，因此发生无法序列化访问的错误。（换一种解释方法：**只要在seriaizabe事务开始到结束之间有其他事务对seriaizabe事务要修改的东西进行了修改并提交了修改，则发生无法序列化访问的错误**。）