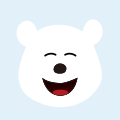
转载数据库阻塞和死锁的区别

[](https://blog.51cto.com/364879551)

[zhxtzf888](https://blog.51cto.com/364879551)0人评论[1610人阅读2018-03-19 17:16:00](javascript:;)

数据库阻塞和死锁在程序开发过程经常出现，怎么样避免呢？下面通过Demo简单模拟下，数据库发生阻塞和死锁的现象：  
一、数据库阻塞：  
数据库阻塞的现象：第一个连接占有资源没有释放，而第二个连接需要获取这个资源。如果第一个连接没有提交或者回滚，

第二个连接会一直等待下去，直到第一个连接释放该资源为止。对于阻塞，数据库无法处理，所以对数据库操作要及时地提交或者回滚。  
Demo:  
--创建表  
create table tb(id int,createtime date);  
--插入测试数据  
insert into tb select 1,sysdate from dual;  
insert into tb select 2,sysdate from dual;  
insert into tb select 3,sysdate from dual;  
commit;

第一个连接，不提交或者回滚：  
SQL> update tb set id=2 where id=1;

1 row updated

第二个连接，一直在运行：  
SQL> update tb set id=2 where id=1;

因为第一个连接占有tb表没有释放资源，而第二个连接一直在等待第一个连接释放该资源。

二、数据库死锁：  
数据库死锁的现象：第一个连接占有资源没有释放，准备获取第二个连接所占用的资源，而第二个连接占有资源没有释放，准备获取第一个连接所占用的资源。这种互相占有对方需要获取的资源的现象叫做死锁。对于死锁，数据库处理方法：牺牲一个连接，保证另外一个连接成功执行。  
Demo:  
--创建测试表t1  
create table t1(id int,createtime date);  
insert into t1 select 1,sysdate from dual;  
insert into t1 select 2,sysdate from dual;  
insert into t1 select 3,sysdate from dual;  
commit;

--创建测试表t2  
create table t2(id int,createtime date);  
insert into t2 select 1,sysdate from dual;  
insert into t2 select 2,sysdate from dual;  
insert into t2 select 3,sysdate from dual;  
commit;

第一个连接,在command窗口中运行：  
begin   
--先修改t1  
update t1 set id=2  
where id=1;  
--等待20s  
dbms\_lock.sleep(20);  
--再修改t2  
update t2 set id=2  
where id=1;  
end;  
/   
运行结果：  
ORA-00060: 等待资源时检测到死锁  
ORA-06512: 在 line 9

第二个连接：

begin   
--先修改t2  
update t2 set id=2  
where id=1;  
--等待20s  
dbms\_lock.sleep(20);  
--再修改t1  
update t1 set id=2  
where id=1;  
end;  
/  
运行结果：  
PL/SQL procedure successfully completed

因为第一个连接占有表t1，想要获取表t2的资源，而第二个连接占有表t2,想要获取表t1的资源，这种互相占有对方想要获取的资源，满足死锁现象。最后第一个连接报异常退出，而第二个连接执行成功。

死锁所在的资源和检测：

在SQL Server的两个或多个任务中，如果某个任务锁定了其他任务试图锁定的资源。会造成这些任务的永久阻塞，从而出现死锁。

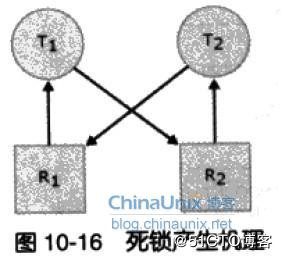
下图为例：

l 事务T1获得了行R1的共享锁。

l 事务T2获得了行R2的共享锁。

l 然后事务T1请求行R2的排它锁，但是T2完成并释放其对R2的共享锁之前被阻塞。

l T2请求行R1的排它锁，但是事务T1完成并释放其对R1持有的共享锁之前被阻塞。

现在T2与T1相互等待，导致了死锁。一般情况下监视器会自动检测并解决这个问题。  


数据库阻塞和死锁的区别  
可以发生死锁的资源：

死锁不仅仅发生在锁资源上面，还会发生在一下资源上：

l 锁。例如页、行、元数据和应用程序上的锁。

l 工作线程。如果排队等待线程的任务拥有阻塞所有其他工作线程的资源，也会导致死锁。

l 内存。当并发请求等待获得内存，而当前的可用内存无法满足其需要时，可能发生死锁。

l 并行查询执行的相关资源。当一个语句用到多个线程运行时，线程之间有可能发生死锁。  
死锁检测：

默认5秒钟搜索SQL Server中的所有任务，检测是否有死锁。如果有，将选择一个作为牺牲品，并返回1205错误。一般是开销最小的事务作为牺牲品。

死锁与阻塞的差别：

阻塞：当一个事务请求一个被其他事务锁定的资源上的锁时，发出请求的事务会一直等待下去，知道该锁被别人释放，自己能申请到位置。

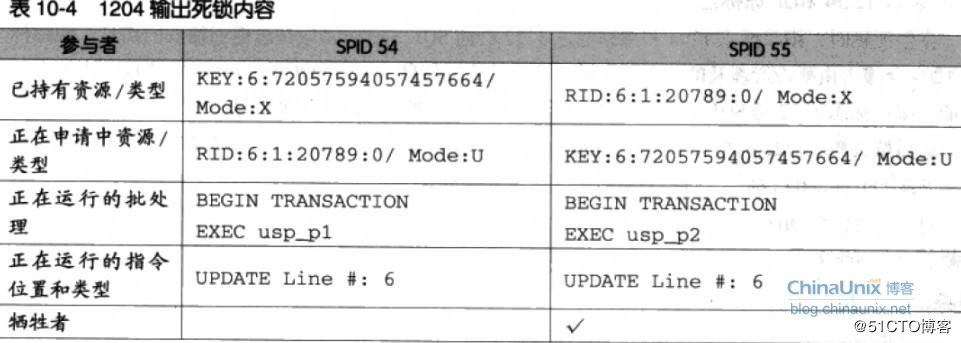
默认情况下除非设置了LOCK\_TIMEOUT，否则事务会一直等待下去。

死锁：两个或多个进程之间的相互等待。但是由于SQL Server有数据库引擎死锁检测方案，至少5秒钟会消除一个现有的死锁。对性能的影响往往没有阻塞严重。  
问题定位：

1、 跟踪标志1204和跟踪标志1222：

打开跟踪的语句：

DBCC TRACEON(1222,-1)

DBCC TRACEON(1204,-1)  


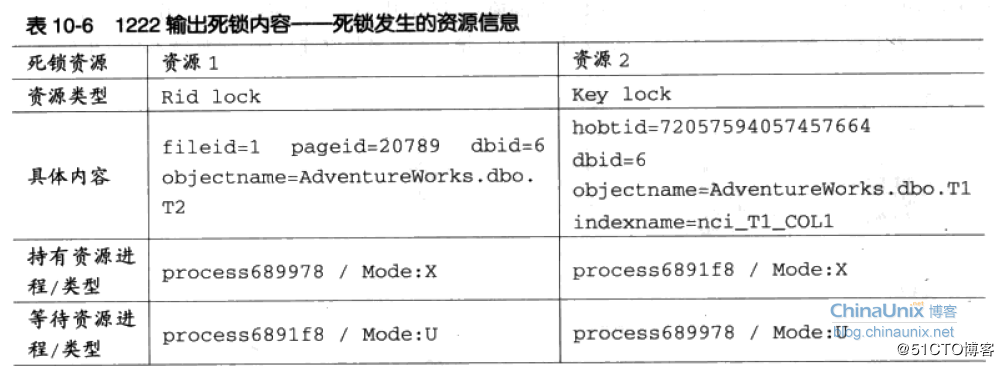
数据库阻塞和死锁的区别

对于1222产生的结果解释：

1、 死锁牺牲的进程：第一句deadlockvictim=processXXXX，中的xxxx就是死锁牺牲品。

2、 死锁发生的进程信息：第二部分的process-list  


数据库阻塞和死锁的区别

3、 发生死锁的资源信息：在结果的resource-list中  


数据库阻塞和死锁的区别

4、 死锁图形事件：

从sqlserver profiler中得到，一般结合1222跟踪标志和sql trace。

首先从errorlog中寻找1222的输出结果，根据输出的时间在跟踪里找到相应的连接。然后分析原因。

解决办法：

尽管死锁不能完全避免，但是可以把机会降到最低：

l 按同一顺序访问对象。

l 避免事务中的用户交互。

l 保持事务简短并处于一个批处理中。

l 使用脚底的隔离级别。

l 调整语句的执行计划，减少锁的申请数目。

按同一顺序访问对象：

如果所有并发事务按同一顺序访问对象，则发生死锁的可能性会降低。

避免事务中的用户交互：

避免编写包含用户交互的事务，因为没有用户干预的批处理的运行速度远快于必须等待用户响应时的查询速度。

保持事务简短并处于一个批处理中：

运行时间越长，等待时间就越长，造成死锁的机会就越高。

使用脚底的隔离级别：

确定事务能否在低隔离级别上运行。尽可能使用较低的隔离级别。

调整语句的执行计划，减少锁的申请数目：

可以从执行计划中找出哪些资源耗得比较多。此时锁的数目也会相应增多。