lock： 锁是Oracle管理共享数据库资源并发访问并防止并发 数据库事务之间“相互干涉”的核心机制之一

|  |
| --- |
| **锁是用来控制并发时的对象完整性** |

是资源或队列，其实是一块内存

1 高级锁（队列所）：dml，table lock，row lock，transaction lock，==》用户可控(

DML锁对象是

o 表

o 表中的数据行

所以说dml锁用来保护表和行数据，防止并发修改

)

2 中级锁： ddl，libarary cache，lock/pin,drop table      ==>用户不可控

DDL锁对象是

o 对象结构！

防止并发ddl操作，

3 低级锁： latch（11g以前），mutex（11g以后）    ==》用户不可控

LATCH(MUTEX)对象是

o 保护内存对象！

Oracle实例SGA对象常见的组件有LIBRARY CACHE，共享池，LOG BUFFER等。

因为这些内存是【共享】的

共享的东西就有很多人抢吧

就会有并发

共享内存就可能会并发的读写

要保证共享内存的一致性，就需要用一种机制来保证

那就是【LATCH】

LATCH是用来保护内存对象不被并发的其他进程修改。

而我们说锁并发控制的本质是排队，先排先得

而LATCH并发控制的本质是【抢】号。拥有的时间极其短暂，循环使用，先抢在排队，先latch 在lock   --房地产绣球

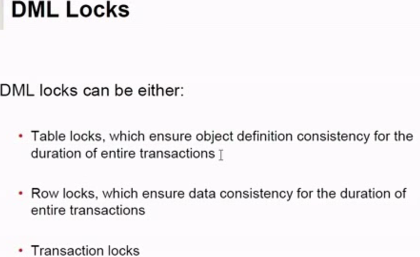
|  |
| --- |
|  |

v$lock  持有锁和请求锁

v$lock\_object  持有锁

v$enqueue\_lock  请求锁

====================================================



update train1 set salary=5000 where employee\_id=22;

==》产生dml lock(table locks,row locks,transaction locks)

**1 table lock：**

TM enqueues :table managed enqueues

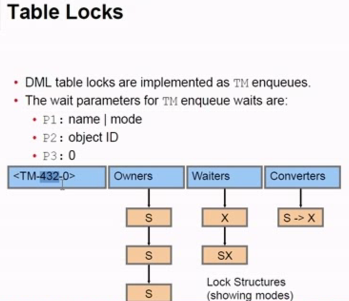




table manager enqueue：以上看到等待队列信息

SQL> desc v$lock;  
 Name                                      Null?    Type  
 ----------------------------------------- -------- ---------------------------  
  
 ADDR                                               RAW(8)   --锁在内存中的地址  
 KADDR                                              RAW(8)  
 SID                                                NUMBER  
 TYPE                                               VARCHAR2(2)   --MR （media recover）保护文件不被恢复 TM,TX  
 ID1                                                NUMBER   --file\_id,object\_id,xid  
 ID2                                                NUMBER  
 LMODE                                              NUMBER   --持有的锁模式  
 REQUEST                                            NUMBER  --请求的锁模式  
 CTIME                                              NUMBER   --持有锁的时间，单位s  
 BLOCK                                              NUMBER   --1阻塞者  
 CON\_ID                                             NUMBER

create table test1 (id number,salary number,name varchar2(22),dat timestamp);

insert into test1 values (1,5000,'aaa','06-jul-05');

commit;

session365

update test1 set salary=6000 where id=1;

session255

update system.test1 set salary=6600 where id=1;

 select sid,type,id1,id2,lmode,request,block from v$lock where sid=365 or sid=255;

       SID TY        ID1        ID2      LMODE    REQUEST      BLOCK  
 ---------- -- ---------- ---------- ---------- ---------- ----------  
        365 AE        133          0          4          0          0  
        255 AE        133          0          4          0          0  
        255 TX     720924         33          0          6          0  
        255 TM      91995          0          3          0          0  
        365 TM      91995          0          3          0          0  
        365 TX     720924         33          6          0          1

type: TM

id1: object id  91995

TM-91995-0

查询dba\_objects



<http://www.itpub.net/thread-1841245-1-1.html>

Oracle的TM锁设置7类锁模式，就是基于并发的进程目的不同。

有7种不同身份和目的的并发用户要排队。

接下来，我们就会来详细分析这7种人 ----世博会参观

|  |
| --- |
| **并发的人目的不同，身份不同，申请的模式也不同，以体现不同的并发能力和权力** |

TM - DML enqueue    （表级管理锁，阻塞了所有dml操作）

TX - Transaction enqueue    （行级事务排他锁，阻塞所有的事务）

UL - User supplied

0号锁  none 没有锁（这种锁模式下可以改变对象结构甚至删除对象）    ==》select （只参观不买）

1号锁  null 易碎解析锁 （也是不买但是属于特殊人群如：老幼病残孕，就是提前通知他们一声）

==》1号锁有两种方式：

o 一些内部操作时会在某些阶段自动获得1号锁

o lock table t in null mode

（library cache lock 1 ==》sql被缓存了就会加1号null锁==》如果sql语句涉及的表结构被修改，则null锁被释放==》重新做sql解析缓存==》1号锁是一个标记。标记这个sql被缓存了）（再比如游标）

2号锁  ss   表里的所有行都可被更改，表结构不能被更改。限制少，并发最高。（试用再买，肯定比只参观不买的待遇高）

==》 自动 select \* from a for update;（10g之前）（查看为了买）

         手动：lock table t in row shared  mode

3号锁  sx   表里面的某几行数据被锁，其他行可以被修改，表结构不能被更改（直接买，权利最大）

==》 自动 select \* from a for update;（10g之后），insert update，delete

         手动：lock table t in 3号  mode

4号锁  s    表里面的所有行数据被锁，不能更改任意行数据，表结构不能被更改，表 read only（观察兼容表与任意带x的都冲突，也就是只读，但是s与s自己兼容，所以可以并发只读）

5号锁  ssx  独有的锁，同一时间只能有一个5号锁，同时不能修改任意行数据，可以读。（单线程，只允许一个人，不是并发通道，ssx与ssx自己不兼容）（

我们如果创建一个5号锁的程序WRITE

那么如果同时两个人调用了程序WRITE

能并发调用WRITE吗？

不能，会堵塞

所以，如果我们有业务需要，需要写一个单用户程序，就可以用lock table在5号模式来实现。

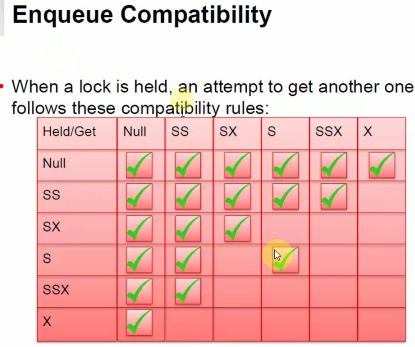
你说Oracle考虑的周全吗）

6号锁  x    排他 ，限制最大，并发最差，同串行（包场，除了null，其他都不兼容！！）

|  |
| --- |
| **从0号锁到6号锁，并发限制的越来越严格，兼容性越来越差** |

锁模式通常是指TM锁！！！锁模式可以对应TM,TX。通常6号锁对象TM,TX    ---珠宝店参观

**对应下面的兼容性enqueues compatibility**



eg：

insert into t .... 语句

先回答，这条语句会产生什么tm锁和tx锁?

3号tm锁

6号tx锁

如果申请3号tm锁成功，就开始执行Insert

然后在该行会生成6号的tx锁

所以先获得表锁，才能去锁定行

先锁定容器，才能去锁定数据

假设

如果两个用户同时insert into t ...

比如A先执行了insert into t ...

A先在t表上获得了3号tm锁

然后开始插入一行记录，在该记录上，A获得了tx 6号锁

接下来

B也要执行insert into t，也要在t表上获得了3号tm锁

问问大家

B能成功获得t表的3号tm锁吗？

A已经在t上有3号锁了

B能申请到t上的3号锁吗？

是能的

否则我们就无法并发差入了

为什么呢？

因为3号锁之间在并发时是兼容的

我们得先来看下表级TM锁的兼容图

那我再来举个例子

如果A用户执行了

update t set ... where id = 1;

那A用户会在表t上获得3号锁，然后在id=1这行设置了6号tx锁

先获得表级锁，再去申请行级锁

行级锁是锁在id=1这行上

此后，B用户也要执行

update t set ... where id = 1;

那B用户也会先去表t上申请SX吧

A持有SX，而B申请SX，是否能成功？

能成功的

此时B用户获得表t的3号锁，接下来，B用户 接着要去获得id=1的tx锁

跟A持有的id=1的tx锁是同一行

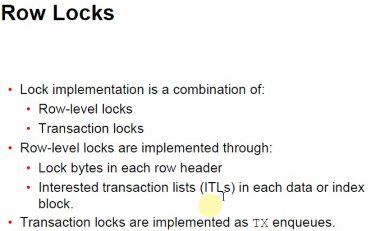
那B想获取id=1的6号x锁能成功吗

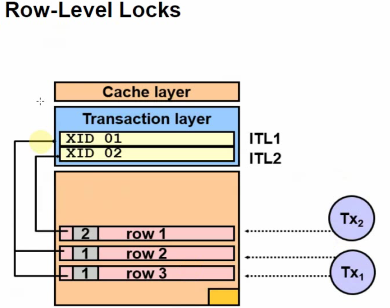
不行

因为行级别上，也要看兼容性的

6号锁跟6号锁是不兼容的

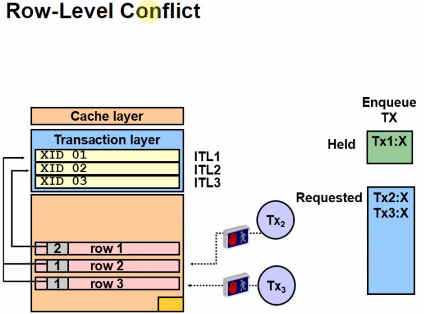
**2 row lock :**



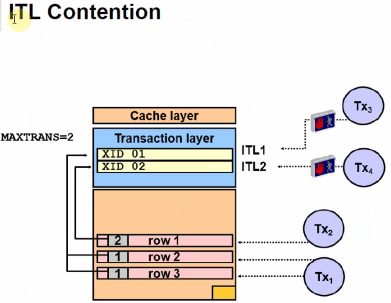


row lock :标记 这一行使用的事务锁类型（事务槽号 2,1,1）！！！没有资源浪费。

TX对应多个row lock（ITL对应多个row）



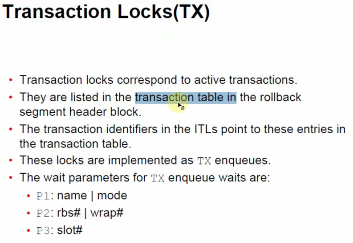
row lock冲突：同一时刻不能修改同一行记录

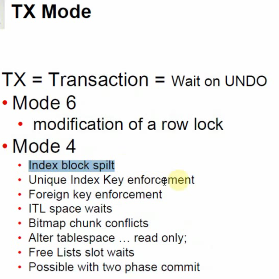


transaction layer中每一个trasaction都会占用一个ITL。每个ITL占用24bytes

9i如果ITL不够（maxtrans=2），就无法分配新的ITL 。10g如果空间不够，也会无法分配新的ITL。

**3 transaction lock：**





事务是行锁的表现，行锁是事务的具体代表。

P1  名字/模式

p2  回滚段/覆盖计数

p3  ITL事务槽位，事务表的第几行

  select sid,type,id1,id2,lmode,request,ctime,block from v$lock where sid=365 or sid=255;

        SID TY        ID1        ID2      LMODE    REQUEST      CTIME      BLOCK

---------- -- ---------- ---------- ---------- ---------- ---------- ----------  
        365 AE        133          0          4          0       4491          0  
        255 AE        133          0          4          0       2441          0  
        255 TX     720924         33          0          6       2421          0  
        255 TM      91995          0          3          0       2421          0  
        365 TM      91995          0          3          0       4393          0  
        365 TX     720924         33          6          0       4393          1

TX-720924-33

tpye:tx

id1: 720924: xid

id2: 33 覆盖次数

select to\_char('720924','xxxxxxxxxxx') from dual;     10>16

TO\_CHAR('720  
 ------------

       b001c

select addr,xidusn,xidslot,xidsqn,status,xid from v$transaction where upper(to\_char('720924','xxxxxxxxxxx')) like '%B001C%';

    XIDUSN    XIDSLOT     XIDSQN STATUS           XID  
 ---------- ---------- ---------- ---------------- ----------------  
         11         28         33 ACTIVE           0B001C0021000000

select addr,xidusn,xidslot,xidsqn,status,xid from v$transaction where xidsqn=id2

来查找到xid的含义（11,28,33）

也可以下手动解析

select to\_char('720924','xxxxxxxxxxx') from dual;     10>16

TO\_CHAR('720  
 ------------

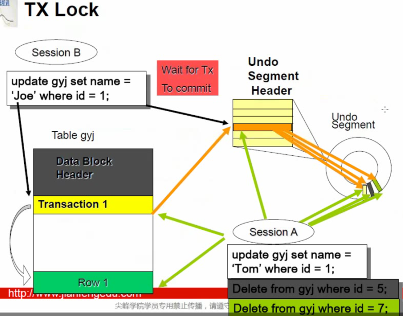
       b001c     ===>高位2个bytes代表回滚段号，低位2个bytes代表事务表槽号

                       ===》0x000b001c===》xidusn 0x000b  ==》11

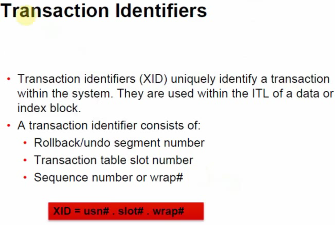
                       ===》0x000b001c===》xidslot 0x001c ==》28

select to\_number('000b ','xxxxxxxx') from dual;

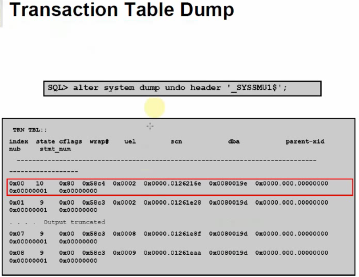
SQL> select to\_number('000b','xxxxxxxx') from dual;  
  
 TO\_NUMBER('000B','XXXXXXXX')  
 ----------------------------  
                           11  
  
 SQL> select to\_number('001c','xxxxxxxx') from dual;  
  
 TO\_NUMBER('001C','XXXXXXXX')  
 ----------------------------  
                           28



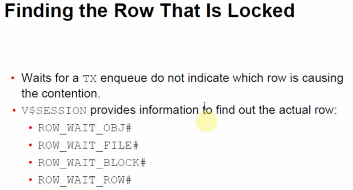
修改同一行，事务是一样的。



xid 串联了data block 和rbu header（rollback unit header）



**处理锁等待！**



更改同样的行，oracle认为事务是一至的！！

  select sid,type,id1,id2,lmode,request,ctime,block from v$lock where sid=365 or sid=255;

        SID TY        ID1        ID2      LMODE    REQUEST      CTIME      BLOCK

---------- -- ---------- ---------- ---------- ---------- ---------- ----------  
        365 AE        133          0          4          0       4491          0  
        255 AE        133          0          4          0       2441          0  
        255 TX     720924         33          0          6       2421          0  
        255 TM      91995          0          3          0       2421          0  
        365 TM      91995          0          3          0       4393          0  
        365 TX     720924         33          6          0       4393          1

以上可以看出session 365阻塞session 255

select \* from v$session where sid in (255,365);

note:

v$session.taddr=v$transaction.addr   ==》获取事务xid

v$session.lockwait=v$transaction.kaddr

v$session.sql\_id=v$sqltext.sql\_id   ==获取sql\_text

commit，rollback

or

alter system kill session '365,51435';

or

kill -9 spid



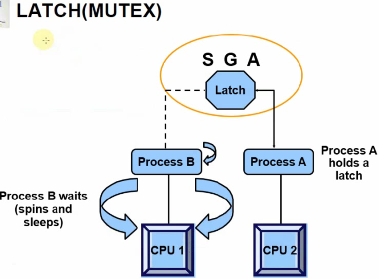
数据字典锁，在share pool中

library cache lock 1号锁=breakable parse locks

3

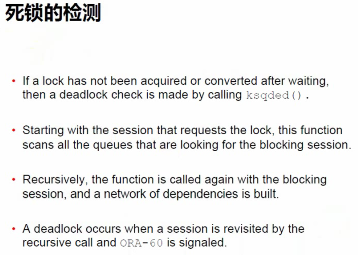
2

1



latch（mutex） 资源是有限的，因为放在sga中，同时消耗cpu。

latch 保护内存结构，参考buffer cache。使用内存资源需要先获取latch保护。如果无法获取就不断spins 大约2000次，还是不能获取就sleeps



ora-60 : 一个主表，一个子表 外键 。阻塞子表！！（4） 外键一定要加索引！！加了索引后，只锁子表对应的行（3）

