<http://zhaizhenxing.blog.51cto.com/643480/134750/>

基本概念

Local Coordinator：在分布事务中，必须参考其它节点上的数据才能完成自己这部分操作的站点。

Global Coordinator：分布事务的发起者，负责协调这个分布事务。

Commit Point Site：在分布事务中，首先执行COMMIT或ROLLBACK操作的站点。一般情况下，应该把存储关键数据的站点作为Commit Point Site。因为Commit Point Site和其它站点不一样，从来不会进入prepared状态，所以不会存在IN-DOUBT事务。

可以设置初始化参数COMMIT\_POINT\_STRENGTH，在分布式事务中，会根据这 个值的大小来确定Commit Point Site，分布事物的状态信息也存在该数据库中。一般将关键的数据库作为commit point site ，commit\_point\_strength值较高的数据库为commit point site，在分布事物中最先提交

分布式提交的3个阶段

分布事物的两阶段提交分三个过程：

1． 准备阶段（PREPARE PHASE）

·本地数据库Global Coordinator向其它数据库发出COMMIT通知

·比较所有数据库的SCN号，将最高的SCN号作为分布事物的全局SCN号

·所有数据库写在线日志

·对分布事物修改的表加分布锁，防止被读写

·各数据库向Global Coordinator发出已经准备好的通知

所有参与分布事物的数据库必须经过上述准备，才能进入下一阶段。

2． 提交阶段（COMMIT PHASE）

·本地数据库Global Coordinator通知commit point site首先提交。commit point site提交后，释放其占有的资源，通知Global Coordinator完成提交

·本地数据库Global Coordinator通知其它数据库提交

·提交节点在日志中追加一条信息，表示分布事物已经完成提交，并通知Global Coordinator。此时所有数据库的数据保持了一致性。

3． 注销阶段（FORGET PHASE）

·本地数据库Global Coordinator通知commit point site所有数据库已经完成提交

·commit point site清除分布事物的记录和状态信息，并通知Global Coordinator

·Global Coordinator清除本地分布事物的记录和状态信息

此时分布事物的两阶段提交全部完成。

如果两阶段提交完成之前，数据库或网络出现异常，应用就会报错，分布事物处于IN\_DOUBT状态。一旦数据库或网络恢复正常,系统（RECO PROCESS）会自动处理IN\_DOUBT状态的分布事物。有些情况需要管理员手工处理IN\_DOUBT状态的分布事物：

·IN\_DOUBT状态的分布事物,将关键表锁住,造成应用不能正常工作

两个重要的视图

DBA\_2PC\_PENDING:列出所有的悬而未决的事务﹐此视图在末填入悬而未决的事务之前是空的﹐解决这后也被清空。

|  |  |
| --- | --- |
| 列名 | 说明 |
| LOCAL\_TRAN\_ID | 本地事务标识﹐格式为integer.integer.ingeger。  当一个连接的local\_tran\_id和global\_tran\_id相同时﹐那么该节点是该事务的全局协调器。 |
| GLOBAL\_TRAN\_ID | 全局事务标识,格式为﹕global\_db\_name.db\_hex\_id.local\_tran\_id,其中db\_hex\_id是用来标识数据库八字符的十六进制数﹐公共事各id在分布式事务的每个节点都是相同的。 |
| STATE | 下图表进行说明 |
| MIXED | “YES”意味着一部分事务已经在一个节点上提交﹐而在另一个节点上被回滚。 |
| TRAN\_COMMENT | 事务的注释﹐或者如果使用了事务命名﹐当事各被提交时﹐事务的名字就会出现在此处 |
| Host | 主机名 |
| Commit# | 已提交的事务的全局提交数 |

DBA\_2PC\_PENDING的STATE列的说明

|  |  |
| --- | --- |
| 列值 | 说明 |
| Connecting | 通常情况下﹐只有全局协调器和本地协调器才使用这个条目﹐节点在能够决定它是否能够准备好之前﹐要收集来自于其它数据库服务的信息。 |
| Prepared | 节点已准好﹐可能或者也可能没有将已准备好的消息通知本地协调器﹐但此时﹐该节点还没有接收到提交的请求﹐仍保持着准许备好的状态﹐控制着提交事务所必需的任何本地资源。 |
| Commited | 节点(任何类型)已经提交了事务﹐但该事务所包含的其它节点可能并没有提交﹐也就是该事务在一个个或多个其它节点上仍然是悬而未决 。 |
| Forced commit | DBA进行判断后﹐可以强行提交未决的事务﹐如果一个事务由DBA在本地节点进行手动提交时﹐产生此项目 |
| Forced abor(rollback) | DBA进行判断后﹐可以强行回滚未决的事务﹐如果一个事务由DBA在本地节点进行手动回滚时﹐产生此项目 |

DBA\_2PC\_NEIGHBORS:列出所有获得的(从远程客户)和送出的(给远程服务器)悬而未决的事务﹐也表示该本地节点是不是事务的提交点站点。

|  |  |
| --- | --- |
| LOCAL\_TRAN\_ID | 同上 |
| IN\_OUT | 获得事务为IN﹐送出事务为OUT |
| Database | 对获得事务来说指本地节点信息的客户数据库的名称﹔对送出的事务来说指用于访问远程服务器上信息的数据库链接的名称 |
| DBuser\_owner | 对获得事务来说指远程数据库链接用于连接的本地账户﹔对于送出事务来说指该数据库链接的拥有者。 |
| INTERFACE | ‘C’代表提交信息﹐’N’表示已准备好状态的一条消息或是一条请求只读提交的请求。  当’IN\_OUT’为OUT时﹐’C’表示该连接的远程的站点是提交点站点,并且知道是提交还是中断。’N’表示本地节点正在通知远程节点﹐说它已准备好。  当’IN\_OUT’为IN时﹐‘C’表示本地节点或送出的远程的一个数据库是提交点站点﹐’N’表示本地节点正在通知远程节点﹐说它已准备好。 |

 处理悬挂事务的一般步骤

1、  检查alert文件，发现类似下面error：

       ORA-1591 "lock held by in-doubt distributed transaction %s"

       ORA-2062 "distributed recovery received dbid x, expected y"

       ORA-2068 "following severe error from %s%s"

2、  确认网络是否正常、dblink是否valid、v$dblink和gv$dblink中查询当前是否在使用分布式事务。

3、  查询视图dba\_2pc\_pending，查询悬挂事务信息：

SELECT LOCAL\_TRAN\_ID, GLOBAL\_TRAN\_ID, STATE, MIXED, HOST, COMMIT#

       FROM DBA\_2PC\_PENDING

       WHERE LOCAL\_TRAN\_ID = '??.';

       如果没有记录，说明RECO进程已经自动处理了该事务。

4、  在所有节点上查询视图dba\_2pc\_neighbors

5、  得到所有节点的COMMIT\_POINT\_STRENGTH值，值最大的为commit point site，即最早提交的点，如果悬挂事务发生在commit point site，则它的state决定了整个分布式事务的状态。悬挂事务是否应该commit force或者是rollback force，由此节点决定。

6、  检查dba\_2pc\_pending的state列，如果是commited，意味着本地数据库提交已经成功。其他节点需要根据本地事务号和最大的commit#进行强制提交。用法如下：

       SVRMGR> COMMIT FORCE 'your local transactionID on this node', 'highest SCN from already committed site';

       SVRMGR> COMMIT FORCE '1.13.5197', '88123887';

7、  如果commit point site的state为commited外的其他状态，则表明commit point site 没有提交成功，分布式事务需要强制回滚。这里不再需要所有节点的最大commit#。用法如下：

       SVRMGR> ROLLBACK FORCE 'your local transactionID on this node';

       SVRMGR> ROLLBACK FORCE '1.13.5197';

8、  清除dba\_2pc\_pending和dba\_2pc\_neighbers的相关记录。一般分布式事务自动恢复后，视图内容会自动清除，如果是手工提交的事务，则需要用dbms\_transaction包手工清除，清除规则如下表所示：

确定何时能使用DBMS\_TRANSACTION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State列 | 全局事务状态 | 本地事务状态 | 通常的动作 | 可选择的动作 |
| Collecting | Rollback | Rollback | 无 | Purge\_lost\_db\_entry(只有当自动回复不能解决事务时) |
| Committed | Committed | Committed | 无 | Purge\_lost\_db\_entry(只有当自动回复不能解决事务时) |
| Prepared | Unknown | Prepared | 无 | 强行提交或回滚 |
| Forced  Commit | Unknown | Committed | 无 | Purge\_lost\_db\_entry(只有当自动回复不能解决事务时) |
| Forced rollback | Unknown | Rollback | 无 | Purge\_lost\_db\_entry(只有当自动回复不能解决事务时) |
| Forced commit | Mixed | Committed | 手动删除不一致性﹐然后使用purge\_mixed |  |
| Forced rollback | Mixed | Rollback | 手动删除不一致性﹐然后使用purge\_mixed |  |

测试记录

¡        设置db1的commit\_point\_strength为1，db2的commit\_point\_strength为2，db2为commit point site。

¡        db1、db2上执行100次insert循环，每次循环用分布式事务插入db1和db2中的测试表。中间reboot db2服务器。此时db1对测试表的查询出现以下错误：

SQL> select count(1) from temp.my\_table;

select count(1) from temp.my\_table

\*

ERROR at line 1:

ORA-01591: lock held by in-doubt distributed transaction 7.30.7415

[oracle@db2 bdump]$ tail -f alert\_ntespay.log

Tue Mar  4 14:14:28 2008

DISTRIB TRAN 1234.4F000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

  is local tran 7.30.7415 (hex=07.1e.1cf7)

  insert pending prepared tran, scn=934346533 (hex=0.37b0ff25)

db1中分布式事务相关的2个视图内容如下：

select a.\* from dba\_2pc\_pending a where LOCAL\_TRAN\_ID='7.30.7415';

         LOCAL\_TRAN\_ID    GLOBAL\_TRAN\_ID STATE       MIXED     ADVICE    TRAN\_COMMENT  FAIL\_TIME       FORCE\_TIME         RETRY\_TIME   OS\_USER  OS\_TERMINAL         HOST        DB\_USER COMMIT#

1       7.30.7415         4660.4F000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  prepared    no                        2008-3-4 14:14:28                 2008-3-4 14:22:56        zhenxingzhai       ZHAIZHENXING         NETEASE\ZHAIZHENXING               934346533

其中，

state有以下几种状态：

Collecting, prepared, committed, forced commit, or forced rollback

mixed表示是否部分提交，部分回滚

advice：

C

for commit,

R

for rollback, else

NULL

select a.\* from dba\_2pc\_neighbors a where LOCAL\_TRAN\_ID='7.30.7415';

      LOCAL\_TRAN\_ID    IN\_OUT    DATABASE       DBUSER\_OWNER     INTERFACE      DBID         SESS#        BRANCH

1       7.30.7415   in      NULLjavaxa.oracle.com        TEMP       N      javaxa\_orcl 1         01000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

此视图说明了数据源1的输入连接信息。因为数据源2不是通过dblink连接的，以此没有出现它的记录。

¡        db2重启后查询my\_tab：

SQL> select count(1) from my\_tab;

  COUNT(1)

----------

        75

¡        因为db2中dba\_2pc\_pending和dba\_2pc\_neighbers中没有记录，并且db2为commit point site，没有记录意味着没有进行任何操作，所以db1应该和db2一样，进行强制rollback。

SQL> conn / as sysdba

Connected.

SQL> rollback force '7.30.7415';

Rollback complete.

SQL> select count(12) from temp.my\_table;

 COUNT(12)

----------

        75

db1的alert日志中显示了可疑事务的回滚过程：

Tue Mar  4 15:14:31 2008

DISTRIB TRAN 1234.4F000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

  is local tran 7.30.7415 (hex=07.1e.1cf7)

  change pending prepared tran, scn=934346533 (hex=0.37b0ff25)

  to     pending forced rollback tran, scn=934346533 (hex=0.37b0ff25)

¡        回滚后，两个视图中的状态更改为如下：

select a.\* from dba\_2pc\_pending a where LOCAL\_TRAN\_ID='9.33.5992';

            LOCAL\_TRAN\_ID    GLOBAL\_TRAN\_ID STATE       MIXED     ADVICE    TRAN\_COMMENT  FAIL\_TIME       FORCE\_TIME         RETRY\_TIME   OS\_USER  OS\_TERMINAL         HOST        DB\_USER COMMIT#

1       7.30.7415         4660.4F000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000  forced rollback  no                        2008-3-4 14:14:28        2008-3-4 15:14:31        2008-3-4 15:20:07        zhenxingzhai         ZHAIZHENXING      NETEASE\ZHAIZHENXING               934346533

select a.\* from dba\_2pc\_neighbors a where LOCAL\_TRAN\_ID='9.33.5992';

      LOCAL\_TRAN\_ID    IN\_OUT    DATABASE       DBUSER\_OWNER     INTERFACE      DBID         SESS#        BRANCH

1       7.30.7415   in      NULLjavaxa.oracle.com        TEMP       N      javaxa\_orcl 1         01000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

¡        去除dba\_2pc\_pending和dba\_2pc\_ neighbors中的记录：

（1） Disable分布式恢复

SQL> ALTER SYSTEM DISABLE DISTRIBUTED RECOVERY;

System altered.

（2）Puege(清空)in-doubt transaction entry：

SQL> exec DBMS\_TRANSACTION.PURGE\_LOST\_DB\_ENTRY('7.30.7415');

PL/SQL procedure successfully completed.

（3）commit;

（4）然后enable 分布式恢复：

SQL> ALTER SYSTEM ENABLE DISTRIBUTED RECOVERY;

分布式事务相关资料

Note:1012842.102

[ORA-2019 ORA-2058 ORA-2068 ORA-2050: Failed Distributed Transactions](http://metalink.oracle.com/)

Note:100664.1

[How to Troubleshoot Distributed Transactions](http://metalink.oracle.com/)

Note:274321.1

[While Trying to Commit or Rollback a Pending Transaction Getting Errors ORA-02058,ORA-01453,ORA-06512](http://metalink.oracle.com/)

Note:126069.1

[Manually Resolving In-Doubt Transactions: Different Scenarios](http://metalink.oracle.com/)