[http://docs.oracle.com/cd/B19306\_01/server.102/b14231/ds\_txnman.htm#i1007609](http://docs.oracle.com/cd/B19306_01/server.102/b14231/ds_txnman.htm" \l "i1007609)

<http://docs.oracle.com/cd/B19306_01/server.102/b14231/ds_txnman.htm>

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

When a transaction is in-doubt, you may need to determine which nodes performed which roles in the session tree. Use to this view to determine:

* All the incoming and outgoing connections for a given transaction
* Whether the node is the commit point site in a given transaction
* Whether the node is a global coordinator in a given transaction (because its local transaction ID and global transaction ID are the same)

1 查看分布式事务，确定global coordinator

col GLOBAL\_TRAN\_ID format a30  
 col HOST format a20  
 col OS\_USER format a20  
 set linesize 200  
 select LOCAL\_TRAN\_ID,GLOBAL\_TRAN\_ID,STATE,to\_char(fail\_time,'dd-mon-yyyy HH24:MI:SS'),mixed,commit#,HOST,db\_user,os\_user from dba\_2pc\_pending;

确定DATABASE 查看使用的dblink，incoming，outgoing，

select LOCAL\_TRAN\_ID,IN\_OUT,DATABASE,DBUSER\_OWNER,INTERFACE,SESS# from dba\_2pc\_neighbors;  

2 手动rollback分布式事务  
 manual rollback  
 rollback force '84.54.7261054';     --dba\_2pc\_pending state会变为forced rollback

3 清理分布式事务从数据字典中

### PURGE\_LOST\_DB\_ENTRY Procedure

When a failure occurs during commit processing, automatic recovery consistently resolves the results at all sites involved in the transaction. However, if the remote database is destroyed or re-created before recovery completes, then the entries used to control recovery in DBA\_2PC\_PENDING and associated tables are never removed, and recovery will periodically retry. Procedure PURGE\_LOST\_DB\_ENTRY enables removal of such transactions from the local site.

Syntax

DBMS\_TRANSACTION.PURGE\_LOST\_DB\_ENTRY (  
 xid VARCHAR2);

 If you got the above ORA-01453 or ORA-06510 errors when trying to purge the transactions, then do the following :  
  
 connect / as sysdba  
 commit; /\* this is to prevent the ORA-01453 in purge\_lost\_db\_entry call \*  
 /  
 exec dbms\_transaction.purge\_lost\_db\_entry( 'trans\_id' );  
  
 =================================================

Usage Notes

WARNING:

PURGE\_LOST\_DB\_ENTRY should only be used when the other database is lost or has been re-created. Any other use may leave the other database in an unrecoverable or inconsistent state.

Before automatic recovery runs, the transaction may show up in DBA\_2PC\_PENDING as state "collecting", "committed", or "prepared". If the DBA has forced an in-doubt transaction to have a particular result by using "commit force" or "rollback force", then states "forced commit" or "forced rollback" may also appear. Automatic recovery normally deletes entries in any of these states. The only exception is when recovery finds a forced transaction which is in a state inconsistent with other sites in the transaction; in this case, the entry is left in the table and the MIXED column has the value 'yes'.

However, under certain conditions, it may not be possible for automatic recovery to run. For example, a remote database may have been permanently lost. Even if it is re-created, it gets a new database ID, so that recovery cannot identify it (a possible symptom is [ORA-02062](http://www.oracle.com/pls/db112/lookup?id=ORA-02062)). In this case, the DBA may use the procedure PURGE\_LOST\_DB\_ENTRY to clean up the entries in any state other than "prepared". The DBA does not need to be in any particular hurry to resolve these entries, because they are not holding any database resources.

The following table indicates what the various states indicate about the transaction and what the DBA actions should be:

Table 152-6 PURGE\_LOST\_DB\_ENTRY Procedure States

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **State of Column** | **State of Global Transaction** | **State of Local Transaction** | **Normal DBA Action** | **Alternative DBA Action** |
| Collecting | Rolled back | Rolled back | None | PURGE\_LOST\_DB\_ENTRY (See Note 1) |
| Committed | Committed | Committed | None | PURGE\_LOST\_DB\_ENTRY (See Note 1) |
| Prepared | Unknown | Prepared | None | FORCE COMMIT or ROLLBACK |
| Forced commit | Unknown | Committed | None | PURGE\_LOST\_DB\_ENTRY (See Note 1) |
| Forced rollback | Unknown | Rolled back | None | PURGE\_LOST\_DB\_ENTRY (See Note 1) |
| Forced commit (mixed) | Mixed | Committed | (See Note 2) |  |
| Forced rollback (mixed) | Mixed | Rolled back | (See Note 2) |  |

NOTE 1:

Use only if significant reconfiguration has occurred so that automatic recovery cannot resolve the transaction. Examples are total loss of the remote database, reconfiguration in software resulting in loss of two-phase commit capability, or loss of information from an external transaction coordinator such as a TP monitor.

NOTE 2:

Examine and take any manual action to remove inconsistencies; then use the procedure PURGE\_MIXED.

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

dba\_2pc\_neighbors  
 INTERFACE      
 C is a commit message; N is either a message indicating a prepared state or a request for a read-only commit.  
 When IN\_OUT is OUT, C means that the child at the remote end of the connection is the commit point site and knows whether to commit or terminate. N means that the local node is informing the remote node that it is prepared.  
 When IN\_OUT is IN, C means that the local node or a database at the remote end of an outgoing connection is the commit point site. N means that the remote node is informing the local node that it is prepared.  
  
 dba\_2pc\_pending  
 STATE can have the following values:  
      \* Collecting  
 This category normally applies only to the global coordinator or local coordinators. The node is currently collecting information from other database servers before it can decide whether it can prepare.  
      \* Prepared  
 The node has prepared and may or may not have acknowledged this to its local coordinator with a prepared message. However, no commit request has been received. The node remains prepared, holding any local resource locks necessary for the transaction to commit.  
      \* Committed  
 The node (any type) has committed the transaction, but other nodes involved in the transaction may not have done the same. That is, the transaction is still pending at one or more nodes.  
      \* Forced Commit  
 A pending transaction can be forced to commit at the discretion of a database administrator. This entry occurs if a transaction is manually committed at a local node.  
      \* Forced termination (rollback)  
 A pending transaction can be forced to roll back at the discretion of a database administrator. This entry occurs if this transaction is manually rolled back at a local node.

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

<<山东省公安厅资源库问题处理(2pc分布式事物、回滚慢)20130920.docx>>