

Oracle Technical Note

분산 데이터베이스

Troubleshooting 시리즈는 필자가 한국오라클 서버지원팀에서 근무하면서 실제 고객들로부터 많은 문의를 받았던 부분들에 대해 단편적인 해결책이 아닌 보다 근본적으로 심도 있게 정리한 것이다. 각 호에서는 각 호마다 다루는 항목에 대한 기본적인 개념과 메커니즘을 설명한 후 업무 과정에서 발생 빈도가 높은 에러를 처리해 나가는 과정을 설명한다.

| 글 | 박경희 | 한국오라클 서버지원팀
| khpark@kr.oracle.com |

지난 호에서는 오라클 메모리 부분에 대해 알아보았고, 이번 호에서는 분산 데이터베이스 부분을 다루어 본다.

분산 데이터베이스는 우리가 자주 접하는 부분임에도 불구하고, 깊이 다루어진 관련 자료의 부재 등으로 실제 에러가 발생하는 경우 적절한 조치가 취해지지 못하고 있다. 또한 단편적으로 에러를 해결한다 하더라도 그 상황에 따른 정확한 개념이 없는 상황에서 취해진 결과로 데이터의 손실이 발생하거나 데이터의 일관성이 깨지는 경우를 종종 볼 수 있다. 심지어는 데이터베이스를 다시 생성하는 곳도 볼 수 있다. 이에 이번 호에서는 분산 데이터베이스의 개념과 사용 방안, 에러 발생시의 조치 방안에 대해 살펴본다.

오라클의 분산 데이터베이스

오라클에서 분산 데이터베이스라 함은 네트워크로 연결된 여러 곳의 컴퓨터에 있는 데이터를 데이터베이스 링크(Database Link)를 사용하여 트랜잭션을 발생시키는 총체적인 단위라 할 수 있다. 즉 데이터베이스 링크를 이용하여 다른 곳에 위치한 데이터베이스의 자료를 조회(Select)하거나 수정(Insert, Update, Delete)하는 기능들을 일컫는다.

더 나아가 오라클에서 지원하는 Readonly Snapshot이나 오라클의 Symmetric Replication(Updatable Snapshot, N-way Master Replication)의 기능들 역시 이 분산 데이터베이스를 이용한 처리 방식이며, 이들의 구현에는 이 데이터베이스 링크가 필수 요소이다.

데이터베이스 링크

데이터베이스 링크란 리모트 데이터베이스의 Path를 정의하는 것이다이를 생성할 때는

```
SQL>CREATE DATABASE LINK test USING 'sales.division.acme.com';
```

의 명령어를 이용하여 접근시 필요한 리모트 데이터베이스를 정의하며, 이를 사용할 때는

```
SQL> SELECT * FROM scott.emp@test
```

처럼 사용한다. 이때 사용되는 sales.division.acme.com은 TNSNAMES.ORA 파일에 연결될 리모트 데이터베이스의 네트워크 정보가 정의된 이름이다.

```
<TNSNAMES.ORA>
```

```
sales.division.acme.com.world =  
(DESCRIPTION =  
(ADDRESS_LIST =  
(ADDRESS =  
(COMMUNITY = tcp.world)  
(DESCRIPTION = (PROTOCOL = TCP)  
(Host = 130.40.11.12)  
(Port = 1521)
```

```

)
(ADDRESS =
(COMMUNITY = tcp.world)
(PROTOCOL = TCP)
(Host = 130.40.11.12)
(Port = 1523)
)
)
(CONNECT_DATA = (SID = SALES )
)
)

```

발생 가능 문제

이 분산 데이터베이스 기능은 '네트워킹'이라는 주요 아키텍처 컴포넌트를 기반으로 이루어지는 만큼 네트워크나 시스템 장애 발생시는 다음과 같은 유형의 문제가 발생할 수 있다.

- Prepare/Commit Process가 진행중에 Failure가 발생하면, Session Tree의 모든 노드에서 정상적인 작업을 종료할 수 없게 될 수도 있다.
- 시스템 장애가 오랜 시간 지속되면, In-doubt 트랜잭션에 의해 Lock된 데이터들을 다른 트랜잭션에서는 사용할 수 없게 된다. (여기에서 언급한 Prepare, Commit, In-doubt 용어에 대해서는 아래에서 다시 설명한다)

또 이러한 상황에 의해 발생하는 오라클 에러는 다음과 같다.

```

ORA-02050: 트랜잭션 ID rolled back,
some remote dbs may be In-doubt
ORA-02051: 트랜잭션 ID Committed,
some remote dbs may be In-doubt
ORA-02054: 트랜잭션 ID In-doubt

```

네트워크나 시스템 장애로 인하여 Prepare/Commit시에 발생하는 위와 같은 에러는 이러한 문제 발생이 복구되는 대로 오라클이 Session Tree의 모든 노드에서 동일한 결과 (All Commit or Rollback) 가 나올 수 있도록 자동 복구를 수행하게 된다. 그러나, 이 분산 작업에 의해 Lock된 데이터를 다른 트랜잭션이 사용하고자 하는 경우에는 무작정 기다리기보다는 적절한 조치를 취해야 할 것이다.

즉, 사용자가 DML 작업을 수행하는 경우, 오라클은 해당 데이터에 대하여 Lock을 획득하고자 한다. 그러나 자신이 필요로 하는 데이터가 다른 트랜잭션에 의해 Lock되어 있는 경우에는 기다리다가 시간이 초과되어 Time-out이 발생한다

- 트랜잭션 시간 초과

사용자가 리모트 데이터베이스에 대하여 Lock을 요구하는 경우, 리모트 데이터베이스에 자신의 로컬 트랜잭션이나 다른 분산 트랜잭션에 의해 이미 Lock이 걸려져 있는 경우 Distributed Lock Timeout 파라미터에 의해 기술된 시간 동안 해당 데이터에 대하여 Lock을 획득하지 못하게 되면, Time-out이 발생하게 된다. 이 경우

해당 트랜잭션은 Rollback이 수행되고 다음의 에러가 발생한다.

ORA-2049 : time-out : distributed transaction waiting for Lock

이 경우 어떤 데이터도 수정되지 않았으므로 어떤 조치를 취할 필요는 없으며, 이후에 다시 트랜잭션을 수행해 보아야 한다. 동일한 현상이 지속적으로 발생하면, 데이터베이스 관리자는 Manual하게 이의 해결 방안을 찾아 보아야 한다.

- In-doubt 트랜잭션에 의한 Lock 현상

로컬 사용자가 로컬 데이터베이스에 대하여 Lock을 요구하는 경우, In-doubt 트랜잭션에 의해 Lock되었다는 메시지를 받을 수 있다. 이러한 경우 Rollback이 수행되고 아래와 같은 에러를 얻게 된다.

ORA-01591 : Lock held by In-doubt distributed transaction ID

이러한 경우, 사용자는 이후에 해당 트랜잭션을 다시 수행해 보아야 하며, 지속적으로 문제가 발생할 때는 데이터베이스 관리자가 문제를 발생시키는 트랜잭션의 ID를 사용하여 해결 방안을 찾아야 한다.

일반적으로 이러한 에러는 드물게 발생하는 사항이며, 네트워크나 시스템 장애가 해결되는 대로 오라클은 자동으로 이의 해결을 위해 동작한다.

Two Phase Commit

Two Phase Commit은 분산 트랜잭션에서 Commit이 발생하는 경우에 사용되는 오라클 메커니즘이며, 오라클은 이를 사용하여 Global Database의 일관성을 유지하게 된다.

다시 말하면, Two Phase Commit 메커니즘은 분산 트랜잭션에 참여하는 모든 노드들에 대하여 'Everything Commit or Rollback' 을 보장하는 데 사용된다.

이때 트랜잭션 종류는 다음 3가지로 나눌 수 있다.

- Local 트랜잭션 : Local 노드만을 처리하는 트랜잭션
- Remote 트랜잭션 : 동일한 하나의 리모트 데이터베이스만을 참조하는 트랜잭션
- Distributed 트랜잭션 : 2개 이상의 서로 다른 데이터베이스의 데이터 수정 작업을 수행하는 트랜잭션

이러한 메커니즘은 100% Transparent하며, 사용자나 애플리케이션 개발자에게 부가적인 작업을 요구하지 않는다.

Prepare와 Commit 단계

분산 트랜잭션의 진행은 아래의 두 단계를 거쳐 수행된다.

1. Prepare Phase : Global Coordinator가 분산 트랜잭션에 참여하는 노드들에 대하여 Prepare하도록 요청한다.
2. Commit Phase : 분산 트랜잭션에 참여하는 노드들이 Prepare되었다는 메시지

를 Global Coordinator에 보내면, Coordinator는 트랜잭션에 대한 Commit을 요구하게 된다. Prepare되지 않았다는 메시지가 오는 경우에는 Rollback을 요구하게 된다.

이처럼 사용자가 Commit 문장을 수행하는 순간 Prepare/Commit 단계가 자동으로 수행되게 된다.

Prepare 단계

분산 트랜잭션의 Commit의 첫 단계인 Prepare 단계에서는 실제로 Commit이 수행되는 것은 아니며, Commit Point Site를 제외한 모든 노드에 대하여 Prepare to Commit을 준비하도록 하는 과정이다.

Prepare는 Commit이나 이후에 수행될지 모르는 Rollback 작업에 대한 정보를 기록하는 것을 말한다. 이처럼 한 노드가 Prepare된 경우 해당 데이터는 Lock된 상태이며 모든 과정 (Prepare/Commit Phase)이 종료되기까지는 데이터를 다른 사용자가 접근할 수 없게 된다.

Prepare하도록 요청받은 노드가 Reply할 수 있는 사항은 다음의 3가지로 국한된다.

- Prepared : 해당 데이터가 수정되었으며, 성공적으로 Prepare가 수행된 경우
- Read-only : 해당하는 데이터가 없어 Prepare가 필요하지 않은 경우
- Abort : Prepare할 수 없는 경우

이 경우 각 노드에서 Prepare 단계시 수행되는 동작을 살펴보면, 다음과 같다.

1. 노드의 Descendant에게 Prepare 하도록 요청
2. 각 노드나 노드의 Descendant에서 해당 트랜잭션에 의해 수정되는 데이터가 있는지를 조사해서 없는 경우 다음 단계인 Commit 단계는 처리하지 않고 Read-only 메시지를 Reply
3. 데이터가 수정된다면, Commit하기 위해 필요한 자원을 할당
4. 트랜잭션에 의해 변화된 사항에 해당하는 엔트리를 로컬 리두 로그에 반영
5. 실패하는 경우에 대비해 Lock을 발생
6. Prepared 메시지를 보낸다. 만약 자신이나 Descendant에서 Prepare가 성공적으로 수행되지 않은 경우에는 트랜잭션이 잡고 있던 Resource를 모두 해지하고, Rollback 작업을 수행함과 동시에 Abort 메시지를 상위 노드에 보낸다. Abort 메시지는 분산 트랜잭션에 참여한 모든 노드들에 보내지며, 모든 노드에서 Rollback 작업이 수행되도록 함으로써 Global Database의 일관성을 유지한다.

위의 기술된 동작들은 분산 트랜잭션이 로컬 트랜잭션처럼 동작하도록 보장한다. Prepared 노드들은 Commit이나 Rollback 메시지가 도착할 때까지 Waiting하게 되며, 노드가 Prepare되었다면, 해당 트랜잭션은 In-doubt 상태에 있게 된다.

Commit 단계

Commit 단계가 수행되기 전에 트랜잭션에 참여하는 모든 노드는 Prepare되어 있어야 한다. Commit 단계는 다음의 순서로 이루어진다.

1. Global Coordinator가 Commit 트랜잭션 메시지를 모든 노드에 보낸다.
2. 각 노드에서는 트랜잭션의 일량 중에서 자신에 해당하는 부분에 대해 Commit을 수행하고 Lock을 해지하며, 로컬 리두 로그에 Commit이 수행되었다는 부가적인 리두 로그 엔트리를 기록한다.

Commit 단계가 종료되면, 분산 시스템의 모든 노드에서의 데이터는 일관성을 유지하게 된다.

용어정리

Session Tree

분산 트랜잭션이 수행되면, 오라클은 트랜잭션에 참여하는 모든 노드의 Session Tree를 구성하게 된다. Session Tree는 Session과 Session Role간의 관계를 Hierarchical하게 구성하는 것을 의미하며, 트랜잭션에 참여하는 모든 노드들은 아래에 기술된 Session Role로 구분된다.

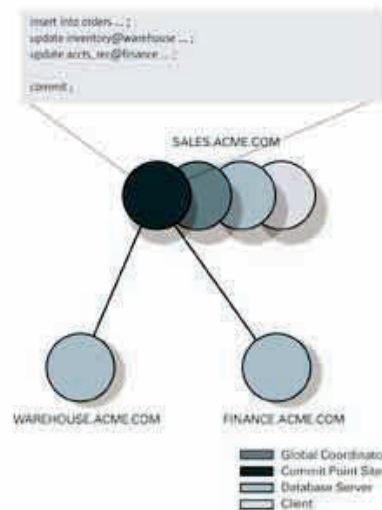


그림1 Session Tree의 예

Clients

다른 노드의 데이터를 참조하는 노드. |그림 1|에서는 sales.acme.com이 Warehouse와 Finance의 데이터를 참조하므로 Client이다.

Server and Database Server

분산 트랜잭션에 의하여 직접 참조되거나, 다른 노드에서 자기 노드의 데이터를 요구하여 분산 트랜잭션에 참여하게 되는 노드. |그림 1|에서는 Warehouse와 Finance가 Database Server가 된다. Sales 노드는 자신의 데이터도 수정하게 되므로 Client와 Database Server Role을 동시에 수행.

Local Coordinator

자신의 분산 트랜잭션 부분을 수행하기 위하여 다른 노드의 데이터를 참조해야만 하는 노드. |그림 1|에서는 Sales 노드가 Local Coordinator이자 Global Coordinator이다. Local Coordinator의 역할은 Descendant 노드와 트랜잭션 Status에 대한 정보를 교환하며, Descendant 노드에 Query를 Pass하거나

Query를 Descendant 노드로부터 받고, 이를 다른 노드로 Pass하고, Query의 결과를 Return 하는 것이다.

Global Coordinator

분산 트랜잭션이 발생한 노드는 Session Tree의 Root 노드가 되며 분산 트랜잭션의 진행중에 다음의 동작을 수행하게 된다.

- 분산 트랜잭션의 SQL문을 Global Coordinator가 직접 Reference하는 노드로 보내주며, Session Tree를 형성한다.
- Commit Point Site를 제외한 노드들에 Prepare를 지시한다.
- 모든 노드에서 Prepare가 성공적으로 수행되면, Global Coordinator는 Commit Point Site에 트랜잭션의 Global Commit을 수행하도록 지시한다.
- Abort 메시지가 오는 경우, Global Rollback을 수행하도록 지시한다.
|그림 1|에서는 Sales 노드가 Global Coordinator의 역할을 한다.

Commit Point Site

Commit Point Site는 Global Coordinator에 의해 지시된 Commit이나 Rollback의 초기화 작업을 수행한다. 시스템 관리자는 항상 하나의 노드를 Commit Point Site로 지정하는 것을 고려하여야 하는데, Commit Point Strength를 사용하여 지정할 수 있다. Commit Point Site는 Critical Data를 저장하는 노드로 선택하는 것이 적절하다.

Commit Point Site는 Prepared State가 될 수 없으며, Commit Point Site에서의 분산 트랜잭션의 결과는 모든 노드에서 Commit이나 Rollback을 결정하게 한다. 그러므로 이 노드가 주요한 데이터를 저장하는 노드이고, 분산 트랜잭션이 실패한다 할지라도 데이터가 In-doubt 상태에 들어가는 경우는 없다.

분산 트랜잭션은 모든 노드가 Prepared되어 있고, Commit Point Site에서 Commit이 수행되었다면 (몇 노드가 아직 Commit되지 않고 Prepared되어 있는 경우에도) Commit이 수행된 것으로 생각한다.

Commit Point Site의 리두 로그는 분산 트랜잭션이 해당 노드에서 Commit되면 즉시 반영된다. 다시 말하면, Commit Point Site에서 Commit이 수행되지 않은 분산 트랜잭션은 Commit되지 않은 것으로 간주된다.

Commit Point Strength

데이터베이스 서버로서 동작을 하는 모든 노드들은 Commit Point Strength가 지정되어야 한다. Commit Point Strength는 해당 노드가 Commit Point Site인지의 여부를 결정하게 된다.

Commit Point Site는 Prepare 단계의 시작 부분에 결정되며, 결정된 후에 Global Coordinator는 참여하는 모든 노드에 대하여 Prepare 메시지를 보내게 된다.

Commit Point Site는 트랜잭션에 대한 상태를 저장하게 되므로, 안정적인 노드로 설정하는 것이 좋다. 예를 들어, Mainframe-based Database Server는 Minicomputer-based Server보다 Commit Point Strength가 큰 것이 좋으며, Minicomputer-based Server는 PC-based Server보다 Commit Point Strength가 큰 것이 좋다.

분산 트랜잭션이 이루어지는 과정을 위한 예제

그러면, 이제 위에서 설명한 분산 트랜잭션을 위한 과정을 예를 들어 살펴보도록 하자.

예를 들어 한 회사에 Sales.acme.com과 Warehouse.acme.com Server가 있으며, Sales Record가 Sales Database에 Insert되는 경우, 관련된 데이터가 Warehouse에 수정되어진다고 가정하자.

발생하는 분산 트랜잭션은 다음의 순서대로 작업이 진행될 것이다.

1. 애플리케이션 프로그램에서 SQL문을 발생. [그림2]

이때 Sales 데이터베이스에서 수행되는 애플리케이션 프로그램이 트랜잭션을 발생시키므로 Sales 노드가 분산 트랜잭션의 Global Coordinator가 된다.

프로그램에서 Sales 데이터베이스에 데이터를 삽입하고 Warehouse 데이터베이스를 수정하므로 Sales와 Warehouse 데이터베이스는 데이터베이스 서버 역할을 가지게 되며, Sales는 Warehouse에 대한 Client Role을 수행한다.

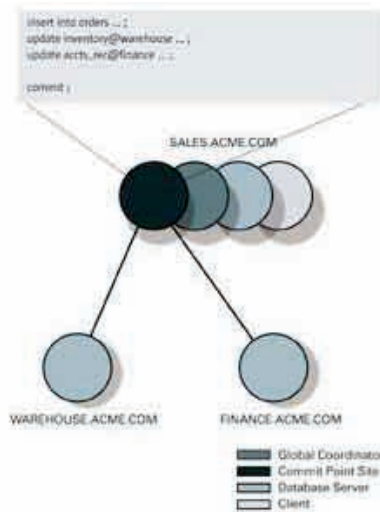


그림2 애플리케이션 프로그램에서 SQL문을 발생

위에 기술된 사항으로 분산 트랜잭션에 대한 Session Tree가 만들어진다 Session Tree의 각 노드는 수행되는 SQL문들에 의해 참조되는 자신의 데이터에 대하여 필요한 Lock을 획득하게 되며, 이는 Prepare/Commit Phase가 종료되기까지 지속된다는 것이다.

2. 애플리케이션에서 Commit 문장을 수행

Commit 문장은 Prepare/Commit 단계 중 Prepare 단계를 시작하게 한다.

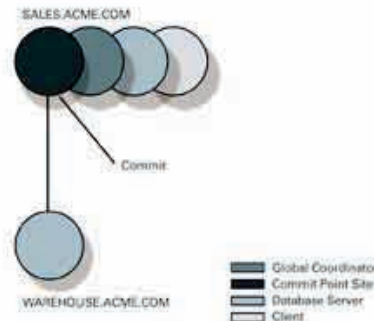


그림3 Global Coordinator의 Commit Point Site 결정

3. Global Coordinator가 Commit Point Site를 결정

Commit Point Strength에 의하여 Commit Point Site가 결정된다.
|그림 3|에서는 Sales 노드를 Commit Point Site로 선정했다.

4. Global Coordinator가 Prepare 메시지를 보냄

Commit Point Site가 결정된 후에 Global Coordinator는 Session Tree에서 직접 참조하는 노드 중 Commit Point Site는 제외한 나머지 노드들에게 Prepare 메시지를 보낸다.

Prepare 메시지를 받은 Warehouse는 Prepare를 시도하게 된다. 트랜잭션 중 해당 노드에 대한 부분을 Commit하는 것이 보장되고, 로컬 리두 로그에 Commit 정보를 기록할 수 있게 되면, Prepare가 성공적으로 수행된 것이다.

|그림 3|에서 Warehouse는 Prepare 메시지를 받게 되고, Prepare 작업을 수행 후, Prepared 메시지를 Reply하게 된다.

만일 Abort 메시지를 보내는 노드가 하나라도 있다면, Global Coordinator는 모든 노드에 트랜잭션을 Rollback하도록 하며, 모든 노드에서 Prepared나 Read-only 메시지를 보내면, Global Coordinator는 Commit Point Site에 Commit을 요청하게 된다. |그림 4|

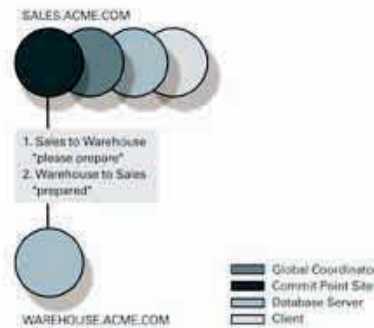


그림4 Global Coordinator가 Prepare 메시지를 보냄

5. Commit Point Site Commit.

Sales 노드는 Warehouse로부터 Prepared 메시지를 받게 되면, Commit Point Site (그림에서는 자기자신)에 트랜잭션의 Commit을 요청하게 된다. Commit Point Site에서는 트랜잭션을 Commit하게 되며, Commit된 정보를 리두 로그에 기록한다. Warehouse 노드가 아직 Commit되기 전이지만 트랜잭션의 결과는 결정된 것이며, Commit이 다소 지연될 수는 있지만, 해당 트랜잭션은 Commit될 것으로 간주된다.

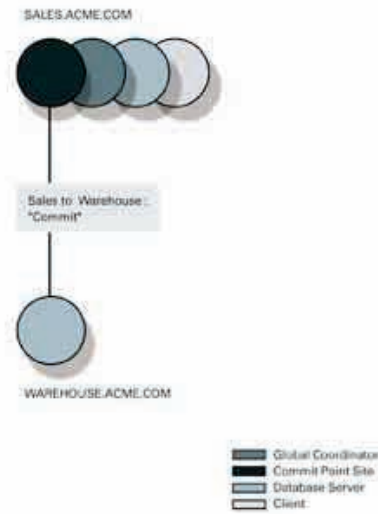


그림5 Commit Point Site가 Global Coordinator에 Commit 여부를 통보

6. Commit Point Site가 Global Coordinator에 Commit 여부를 통보

Commit Point Site는 Global Coordinator에 Commit되었다는 사실을 통보하게 된다. Commit Point Site는 분산 트랜잭션에 참여하는 모든 노드에서 Commit되었다는 정보를 Global Coordinator로부터 확인받기 전까지는 해당 트랜잭션이 Commit되었다는 정보를 유지하게 된다. Commit Point Site로부터 Commit되었다는 메시지를 받은 Global Coordinator는 모든 노드에 Commit을 수행하도록 메시지를 보내게 되며, 각 노드에서 Commit이 수행되면 각 노드의 Lock은 해제된다. | 그림 5|

7. Global Coordinator와 Commit Point Site에서 Commit을 종료

Global Coordinator와 모든 참조되는 노드가 정상적으로 Commit을 수행하게 되면, Global Coordinator는 Commit Point Site에 Commit되었다는 사실을 통보하게 된다.

Commit Point Site는 Global Coordinator로부터 메시지를 받은 후 분산 트랜잭션에 대한 상태 정보를 지우게 되며, Global Coordinator에게 이 사실을 통보한다. Global Coordinator가 이러한 사실을 통보받게 되면, Global Coordinator도 트랜잭션에 대한 정보를 지운 후 트랜잭션을 종료하게 된다.

모든 Commit이 수행되는 트랜잭션들은 (SCN System Change Number) 을 가지게 되는데, 분산 환경에서도 SCN이 적절하게 정의된다. 이는 Global Distributed Read-consistency를 유지하게 하며, 필요한 경우 Global Distributed Time-based Recovery를 수행 가능하게 한다.

오라클은 Prepare Phase에 트랜잭션에 참여하는 모든 노드들 중에서 가장 높은 SCN을 결정하게 되며, Commit Point Site에서 정해진 SCN을 가지고 Commit을 수행하게 된다. Commit SCN은 Commit되어야 하는 모든 노드로 보내진다.

처리 방안

데이터베이스 관리자가 처리하여야 하는 In-doubt 트랜잭션은 Ora-1591 에러를 발생시킨다. 즉 In-doubt 트랜잭션이 Lock을 잡고 있는 데이터를 다른 트랜잭션이 참조하고자 하는 경우이다. In-doubt 트랜잭션은 자신이 사용중인 Rollback Segment를 다른 트랜잭션이 사용하지 못하도록 한다.

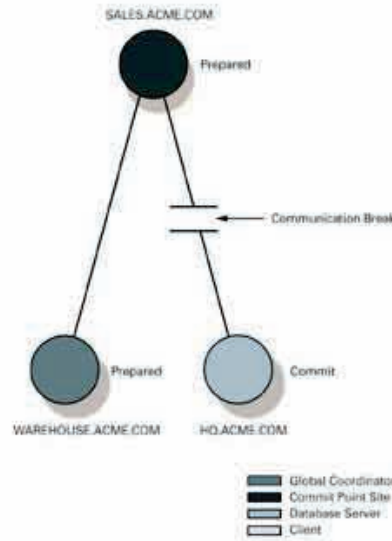


그림6 분산 트랜잭션의 Commit 동안에 Failure가 발생한 경우

DBA_2PC_PENDING과 DBA_ROLLBACK_SEGS View를 통하여 확인해 보면, In-doubt Local 트랜잭션 ID의 첫번째 부분이 사용중인 Rollback Segment ID를 지정하고 있다. 실패한 원인을 해결하는 시간이 많이 소요될 경우, 강제로 Commit이나 Rollback할 수 있다.

위에 기술된 조건이 만족하지 않는 상황이라면, 오라클이 자동으로 해당 트랜잭션을 복구하도록 허용하는 것이 좋으며, Manual하게 작업을 시도하는 경우에는 아래와 같은 정보를 먼저 확인해 보아야 한다.

1. 트랜잭션이 Commit되거나 Rollback된 노드가 있는지 확인한다. 이미 Commit되거나 Rollback된 노드가 있다면, 이의 결과대로 다른 노드에서도 Commit/Rollback을 수행하면 된다.
2. DBA_2PC_PENDING.TRAN_COMMENT 컬럼의 정보를 확인하여 본다. Tran_comment column은 Commit시 Comment를 추가하는 경우 삽입되며, 트랜잭션의 발생 노드와 트랜잭션 종류 등을 기록할 수 있다.

```
Commit comment 'Finance/Accts_pay/Trans_type 10B';
```

3. DBA_2PC_PENDING.ADVISE column의 정보를 확인하여 본다. 애플리케이션에서 Alter Session Command를 사용하여 각 노드에서 해당 트랜잭션을 어떻게 취급할 것인지에 대해 기술하여 줄 수 있다.

```
alter session advise Commit;
insert into emp@hq .....; /* advice to Commit at HQ */
alter session advise Rollback;
delete from emp@sales .....; /* advice to roll back at SALES */
alter session advise nothing;
```

분산 트랜잭션의 Commit 동안에 Failure가 발생한 경우, Manual하게 Commit이나 Rollback을 수행하기 전에 어떻게 정보를 확인하여 나가는지에 대하여 살펴보자.

|그림 6 |은 Prepare 단계는 종료된 상태에서 Commit 단계를 수행할 때 네트워크 장애가 발생하여 Commit Point Site는 Commit이 되었으나, Commit 메시지를 Global Coordinator에 Reply하지 못하고 있는 경우이다.

복구 예제

Warehouse Database 관리자의 입장에서 생각해 보면, In-doubt 트랜잭션에 의해 Lock되어 데이터가 있으며 다른 트랜잭션에 의해 빈번하게 해당 데이터가 요구되는 경우, In-doubt 트랜잭션이 Commit되거나 Rollback되기까지는 해당 데이터를 액세스할 수 없다. 또한 Failure의 해결에 많은 시간이 소요될 것이라고 하면, Manual하게 작업을 수행할 것을 고려해 보아야 한다. 그리고 Manual하게 작업을 수행하고자 한다면 아래의 절차를 따라 작업을 진행한다.

1. 에러 메시지를 기록한다.
2. In-doubt 트랜잭션의 Global 트랜잭션 ID와 부가적인 정보들을 얻기 위해 DBA_2PC_PENDING View를 Select한다.
3. In-doubt 트랜잭션이 해결된 노드를 찾기 위해 Session Tree를 추적해 보아야 하는데, 이를 위하여 DBA_2PC_NEIGHBORS View를 Select한다.
4. 정상적으로 Communication이 복구된 상황에서 Mixed Flag를 확인한다.

Step 1: Record User Feedback

In-doubt 트랜잭션에 의해 Lock된 데이터를 로컬 데이터베이스 사용자가 사용하고 자 하는 경우 아래와 같은 에러가 발생하게 되는데, 이를 먼저 기록하여 둔다.

```
ORA-01591: Lock held by In-doubt distributed 트랜잭션 1.21.17
```

Step 2: Query DBA_2PC_PENDING

In-doubt 트랜잭션에 대한 정보를 얻기 위하여 dba_2pc_pending을 Select한다.

```
SELECT * FROM sys.dba_2pc_pending  
WHERE local_tran_id = '1.21.17';
```

Select 문은 Warehouse 노드에서 수행되었으며, 결과는 아래와 같다.

Column Name	Value
LOCAL_TRAN_ID	1.21.17
GLOBAL_TRAN_ID	SALES.ACME.COM.55d1c563.1.93.29
STATE	Prepared
MIXED	no
ADVICE	
TRAN_COMMENT	Sales/New Order/Trans_type 10B
FAIL_TIME	31-MAY-91

FORCE_TIME	
RETRY_TIME	31-MAY-91
OS_USER	SWILLIAMS
OS_TERMINAL	TWA139:
HOST	system1
DB_USER	SWILLIAMS
COMMIT#	

Global 트랜잭션 ID는 분산 트랜잭션에 참여한 모든 노드에서 동일하며, 아래와 같은 형태를 가진다.

`Global_database_name.hhhhhhhh.local_트랜잭션_id`

Global_database_name은 Global Coordinator의 Database Name이며, hhhhhhhh는 Global Coordinator의 Internal Database ID이다.

그리고 Local_트랜잭션_id는 Global Coordinator에 Assign된 Local 트랜잭션 ID이다. 위와 같이 구성되어 있으며, Global 트랜잭션 ID의 마지막 부분과 Local 트랜잭션 ID가 일치하면, 해당 노드가 Global Coordinator가 된다. 위의 결과에서는 일치하고 있지 않으며 Sales 노드가 Global Coordinator인 것을 알 수 있다.

State Column은 Warehouse 노드가 Prepared State인 것을 보여주며, Commit이나 Rollback Command를 기다리고 있음을 알 수 있다.

Step 3: Query DBA_2PC_NEIGH-BORS

이 단계에서는 Session Tree를 탐색하게 되며, Commit Point Site를 찾아 동일하게 Local 트랜잭션을 처리하도록 하는 데 목적이 있다.

DBA_2PC_NEIGHBORS View는 In-doubt 트랜잭션과 관련된 Connection에 대한 정보를 제공한다. Select의 결과에서 in_out column의 값에 따라 해당 노드의 Role을 결정할 수 있다.

- in 값을 가지는 경우: Query가 수행된 노드는 다른 노드의 Database Server가 되며, Database Column이 Client Database Name을, dbuser_owner Column은 In-doubt 트랜잭션에 대한 Database Link의 Local Account를 가리키게 된다.
- out 값을 가지는 경우: Query가 수행된 노드는 다른 Server에 대한 Client가 된다. 이 경우 Database Column이 Remote 노드에 Connect하는 Database Link Name 값을 가지며, dbuser_owner Column은 In-doubt 트랜잭션에 대한 Database Link의 Owner를 지정하게 된다.
- 부가적으로 Interface Column은 해당 노드가 Commit Point Site인지를 확인하여 준다.

```
SELECT * FROM sys.dba_2pc_neighbors
WHERE local_tran_id = '1.21.17'
ORDER BY sess#, in_out;
```

Column Name	Value
LOCAL_TRAN_ID	1.21.17
IN_OUT	in
DATABASE	SALES.ACME.COM
DBUSER_OWNER	SWILLIAMS
INTERFACE	N
DBID	000003F4
SESS#	1
BRANCH	0100

SALES.ACME.COM의 조사 결과

```
SELECT* FROM sys.dba_2pc_pending
WHERE Global_tran_id = 'SALES.ACME.COM.55d1c563.1.93.29';
```

Column Name	Value
LOCAL_TRAN_ID	1.93.29
GLOBAL_TRAN_ID	SALES.ACME.COM.55d1c563.1.93.29
STATE	Prepared
MIXED	no
ADVICE	
TRAN_COMMENT	Sales/New Order/Trans_type 10B
FAIL_TIME	31-MAY-91
FORCE_TIME	
RETRY_TIME	31-MAY-91
OS_USER	SWILLIAMS
OS_TERMINAL	TWA139:
HOST	system1
DB_USER	SWILLIAMS
COMMIT#	

```
SELECT * FROM dba_2pc_neighbors
WHERE Global_tran_id =SALES.ACME.COM.55d1c563.1.93.29?br>?
ORDER BY sess#, in_out;
```

Column Name	Value
LOCAL_TRAN_ID	1.93.29
IN_OUT	OUT
DATABASE	WAREHOUSE.ACME.COM
DBUSER_OWNER	SWILLIAMS
INTERFACE	N
DBID	55d1c563
SESS#	1
BRANCH	1
LOCAL_TRAN_ID	1.93.29
IN_OUT	OUT
DATABASE	HQ.ACME.COM

```

DBUSER_OWNER      ALLEN
INTERFACE         C
DBID              00000390
SESS#             1
BRANCH

```

HQ.ACME.COM의 조사 결과

```

SELECT * FROM dba_2pc_pending
WHERE Global_tran_id = 'SALES.ACME .COM.55d1c563.1.93.29';

```

Column Name	Value
LOCAL_TRAN_ID	1.45.13
GLOBAL_TRAN_ID	SALES.ACME.COM.55d1c563.1.93.29
STATE	COMMIT
MIXED	no
ACTION	
TRAN_COMMENT	Sales/New Order/Trans_type 10B
FAIL_TIME	31-MAY-91
FORCE_TIME	
RETRY_TIME	31-MAY-91
OS_USER	SWILLIAMS
OS_TERMINAL	TWA139:
HOST	system1
DB_USER	SWILLIAMS
COMMIT#	129314

Step 4: Check for Mixed Outcome

Manual하게 트랜잭션을 Commit하거나 Rollback한 후에도 Pending 트랜잭션 테이블에는 해당 정보가 남아 있으며, 트랜잭션의 상태가 Forced Commit이나 Forced Abort로 변하게 된다. Failure가 정상적으로 복구된 후에는 RECO Process가 트랜잭션의 Global 결과를 점검하게 되며, 사용자가 잘못된 방식으로 Local 트랜잭션을 Force하였다면 MIXED Column의 값을 Yes로 변환 후, 해당 Row를 삭제하지 않고 유지한다. 이 경우는 일관성이 유지되지 않는 데이터가 존재하는 경우이므로 이에 대한 적절한 조치가 요구되며, 조치 후에는 Pending 트랜잭션 테이블에서 불필요한 Row를 삭제해 주어야 한다.

Pending 트랜잭션 테이블(DBA_2PC_PENDING)

오라클은 분산 트랜잭션에 대한 정보를 유지하기 위하여 Pending 트랜잭션 Table을 유지하며, 이의 내용은 dba_2pc_pending View를 통하여 확인할 수 있다.

Pending 트랜잭션 Table에 Entry를 가진 각 트랜잭션은 다음과 같은 Status로 구분된다.

```
DBA_2PC_PENDING.STATE 컬럼의 값에 따라 구분
collecting
Prepared
Committed
forced Commit
forced abort (rollback)
```

In-doubt 트랜잭션을 강제로 Commit, Rollback하기

```
Commit force 'transaction_id';
Rollback force 'transaction_id';
```

를 사용하여 강제로 Commit이나 Rollback이 가능하다. 또한

```
Commit Force 'Global_id', scn;
```

처럼 SCN을 사용하여 Commit을 수행할 수 있다.

Reco Process 관련 파라미터

RECO Process는 분산 트랜잭션의 수행중 발생한 장애를 자동으로 해결하는 작업을 수행하며, Exponential 한 시간 간격으로 In-doubt 트랜잭션을 검사하면서 복구를 시도하게 된다. In-doubt 트랜잭션이 해결되면, 각 Database의 Pending 트랜잭션 테이블의 해당 Row는 자동으로 삭제된다.

```
Alter system disable distributed recovery;
```

Alter system enable distributed recovery; Command를 사용하여 Enable/Disable이 가능하다.

관련 파라미터는 다음과 같다.

- DISTRIBUTED_RECOVERY_CONNECTION_HOLD_TIME (default 200 seconds): 분산 트랜잭션이 실패하는 경우 Connect를 유지하고 있는 시간을 Setting하는 Parameter
- DISTRIBUTED_TRANSACTIONS: 한 Database가 참여할 수 있는 분산 트랜잭션의 Max 값
- DISTRIBUTED_LOCK_TIMEOUT (default 60 seconds): Locked Resource에 대한 분산 트랜잭션의 Waiting Time

관련 패키지

?/rdbms/admin/dbmsutil.sql을 실행하여 생성되는 DBMS_TRANSACTION의 패키지를 이용하여 In-doubt 트랜잭션의 처리도 가능하다. 즉 한 리모트 데이터베이스를 재생성해야 하는 경우는 Local 데이터베이스에서

```
sql>execute sys.dbms_transaction.purge_lost_db_entry(" ");
```

를 실행하여 In-doubt 트랜잭션을 제거할 수 있다 .

분산 데이터베이스 운용을 위해 유용한 Procedure와 Function은 다음과 같다.

```
DBMS_TRANSACTION.COMMIT_COMMENT  
DBMS_TRANSACTION.COMMIT_FORCE  
DBMS_TRANSACTION.COMMIT  
DBMS_TRANSACTION.ROLLBACK  
DBMS_TRANSACTION.ROLLBACK_SAVEPOINT  
DBMS_TRANSACTION.ROLLBACK_FORCE  
DBMS_TRANSACTION.PURGE_MIXED  
DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY  
DBMS_TRANSACTION.LOCAL_TRANSACTION_ID  
DBMS_TRANSACTION.STEP_ID
```



한국오라클(주)

서울특별시 강남구 삼성동 144-17
삼화빌딩
대표전화 : 2194-8000
FAX : 2194-8001

한국오라클교육센터

서울특별시 영등포구 여의도동 28-1
전경련회관 5층, 7층
대표전화 : 3779-4242~4
FAX : 3779-4100~1

대전사무소

대전광역시 서구 둔산동 929번지
대전둔산사학연금회관 18층
대표전화 : (042)483-4131~2
FAX : (042)483-4133

대구사무소

대구광역시 동구 신천동 111번지
영남타워빌딩 9층
대표전화 : (053)741-4513~4
FAX : (053)741-4515

부산사무소

부산광역시 동구 초량동 1211~7
정암빌딩 8층
대표전화 : (051)465-9996
FAX : (051)465-9958

울산사무소

울산광역시 남구 달동 1319-15번지
정우빌딩 3층
대표전화 : (052)267-4262
FAX : (052)267-4267

광주사무소

광주광역시 서구 양동 60-37
금호생명빌딩 8층
대표전화 : (062)350-0131
FAX : (062)350-0130

고객에게 완전하고 효과적인
정보관리 솔루션을 제공하기 위하여
오라클사는 전 세계 145개국에서
제품, 기술지원, 교육 및
컨설팅 서비스를
제공하고 있습니다.

<http://www.oracle.com/>
<http://www.oracle.com/kr>