락은 도대체 왜거나?

데이터의 무결성을 보장하기 위해 SQL Server에서는 데이터에 Lock을 건다.

1) 교착 상태(DEADLOCK) 개요   
  
교착상태 란, 둘 이상의 스레드 간에 특정 자원(테이블,행..) 대한 양방향 참조(종속) 관계가 발생한 경우를   
말한다. 교착 상태는 주로 단순한 RDBMS 보다는, MULTI THREAD 가 가능한 시스템에서 발생할 가능성이   
더 크다.   
특정 트랜잭션의 스레드는 하나의 이상의 자원에 잠금을 얻을 수 있다. 지금까지 보았던 것처럼 잠겨진   
자원을 다른 트랜잭션의 스레드가 엑세스하려 한다면, 대상 리소스가 해제될 때까지 기다려야 하는   
BLOCKING 이 발생한다. 이 때 대기 중인 스레드는 리소스를 소유한 스레드에 대해 해당 리소스에 대한   
종속 관계를 갖고 있다고 말한다.   
  
예를 들어, 트랜잭션 1을 실행하는 스레드 T1에는 EMP 테이블에 대한 배타적 잠금이 있고, 트랜잭션 2를   
실행하는 스레드 T2는 CUST 테이블에 대한 배타적 잠금이 있다.   
이때 T2는 EMP에 대해 잠금을 요청할 수 있다. 그러나 T1이 현재 배타적 잠금을 가지고 있으므로   
BLOCKING 이 발생한다. T2는 차단되고 T1이 완료되기를 기다릴 수밖에 없는 상황이다.   
이것은 일반적인 BLOCKING 이다.   
그러나, 이런 상황에서 T1이 CUST 테이블에 잠금을 요청하려 하면, 역시 T2가 잠궈 놓았으므로   
BLOCKING 이 발생한다. 일반적인 BLOCKING 이라면, 잠금을 건 해당 트랜잭션이 COMMIT/ROLLBACK   
전까지만 기다리면 된다. 하지만 지금은 일반적인 BLOCKING 이 아니다.   
이런 때는 서로 잠금을 해제할 수 없으며, 또한 커밋하거나 롤백할 수도 없다.   
이렇게 서로 잠금을 소유하고 있는 두 트랜잭션의 스레드에서, 상대방이 소유한 자원을 원하는 경우에는   
교착 상태가 발생한다. 두 스레드 모두 트랜잭션을 커밋하거나 롤백할 때까지 자신이 소유한 리소스를   
해제할 수 없고, 또 다른 트랜잭션이 소유한 리소스를 대기하고 있으므로 트랜잭션을 커밋하거나   
롤백할 수도 없다.   
  
<SESSION1>   
BEGIN TRAN   
UPDATE EMP   
SET SAL = 40000   
WHERE EID = 1   
DBCC OPENTRAN   
EXEC SP\_LOCK 51   
  
실행한 SP\_LOCK명령에 의해, 현재 해당 키인 EID 1번에 대해 X 락이 걸려 있는 것이 확인된다.   
  
<SESSION2>   
BEGIN TRAN   
UPDATE CUST   
SET CTEL = '032-333-9999'   
WHERE CID = 2   
DBCC OPENTRAN   
EXEC SP\_LOCK 52   
  
역시 키인 CID 1번에 대해 X 락이 걸려 있다.   
이번엔 첫 번째 트랜잭션에서 두 번째 트랜잭션에서 배타적 잠금을 설정한 CID 1번을 수정시도 해보자.   
SQL Server 가 잠금을 요청할 것이나, 현재 배타적 잠금이 벌써 두 번째 트랜잭션에 의해 잠금이   
설정된 상태이므로 BLOCKING 이 발생한다. 이것은 일반적인 BLOCKING 이다.   
그러므로 계속 대기상태에 접어든다.   
  
<SESSION1>   
BEGIN TRAN   
UPDATE EMP   
SET SAL = 40000   
WHERE EID = 1   
DBCC OPENTRAN   
EXEC SP\_LOCK 51   
DELETE CUST   
WHERE CID = 2   
  
  
SELECT @@TRANCOUNT   
ROLLBACK TRAN   
  
현재 BLOCKING 이 설정되어 있다.   
아래의 쿼리를 보면, 이번에는 두 번째 트랜잭션에서 다시 첫 번째 트랜잭션에서 배타적 잠금을 걸고   
있는 자원인 EMP 테이블의 EID 값에 대해서 삭제작업을 시도해보자. 또 BLOCKING 이 발생한다.   
그러나 이번에는 DEADLOCK 이 발생하게 되고, 그리고 두 트랜잭션 중 하나만 계속 진행 중인   
상태로 남고 나머지 하나는 자동으로 ROLLBACK 된다.   
  
<SESSION2>   
BEGIN TRAN   
UPDATE CUST   
SET CTEL = '032-333-9999'   
WHERE CID = 2   
DBCC OPENTRAN   
EXEC SP\_LOCK 52   
DELETE EMP   
WHERE EID = 1   
ROLLBACK   
  
위에서 보여주는 바로는 두 번째 트랜잭션이 에러번호 1205번을 출력하고, 취소되었다.   
그리고 첫 번째 트랜잭션은 여전히 진행 중이다.   
  
<SESSION1>   
BEGIN TRAN   
UPDATE EMP   
SET SAL = 40000   
WHERE EID = 1   
DBCC OPENTRAN   
EXEC SP\_LOCK 51   
DELETE CUST   
WHERE CID = 2   
  
SELECT @@TRANCOUNT   
ROLLBACK TRAN   
  
이렇게 BLOCKING 상태가 계속 진행되지 않고 하나의 트랜잭션이 바로 취소되는 이유는, DEADLOCK   
상태를 취소하지 않으면 두 트랜잭션 모두 자신의 잠금을 풀 수 없기 때문에 잠금상태가 영원히   
지속되기 때문이다. DEADLOCK 상태는 트랜잭션이 스스로 잠금을 풀 수 없으므로, SQL Server 가   
자동으로 둘 중 하나의 잠금을 풀고 트랜잭션을 취소시키는 것이다.   
  
여러분들이 해야할 일은 이렇게 DEADLOCK 일어나 트랜잭션 중 하나가 취소된 것을 감지하고   
해당 트랜잭션을 다시 실행하거나 취소하는 처리이다. DEADLOCK 감지는 @@ERROR 1204번으로 한다.   
어떤 트랜잭션에서 @@ERROR 1205번 에러가 발생했다면 데드락이 발생했다는 것이고,   
해당 트랜잭션을 제어해주는 로직을 여러분들이 해당 트랜잭션 내에 추가해야 하는 것이다.   
  
/\* 참고   
교착 상태는 종종 일반적인 BLOCKING 과 혼동된다. 한 트랜잭션이 다른 트랜잭션이 원하는   
리소스를 잠그고 있으면 두 번째 트랜잭션이 해당 잠금이 해제되기를 기다린다. 기본적으로,   
LOCK\_TIMEOUT이 설정되지 않는 한, SQL Server 트랜잭션 시간은 제한되지 않는다.   
이 경우 두 번째 트랜잭션이 차단되지만 교착 상태는 아니다.   
\*/   
  
  
4) 교착상태 분석   
  
추적 플래그 1204 사용   
  
이것은 명령프로프트에서 실행하는 명령어이다. SQL Server 시작 옵션 중의 하나인 /T 옵션(대문자)를   
사용해서 SQL Server를 시작하게 되면 볼 수 있다.   
교착 상태에서 이 추적 플래그 1204는 대기 상태의 스레드, 이 스레드가 대기하고 있는 리소스,   
리소스 간의 종속 관계 주기를 나타낸다.   
  
추적 플래그 1204 보고서 용어   
  
추적 정보 1204는 관련된 리소스에 따라 다른 정보를 반환하지만 일반적으로 보고서에는   
다음과 같은 용어가 포함된다 .   
  
- Node:x   
교착 상태 체인에서 항목 번호(x)를 표시한다.   
  
- Lists   
잠금 소유자가 다음 Grant, Convert 및 Wait 목록의 일부일 수 있다.   
  
- Grant List   
리소스의 현재 소유자를 열거한다.   
  
- Convert List   
잠금을 더 높은 수준으로 변환하려는 현재 소유자를 열거한다.   
  
- Wait List   
리소스에 대한 현재 새 잠금 요청을 열거한다.   
  
- SPID: x ECID: x   
병렬 프로세스의 경우 시스템 프로세스 ID 스레드를 확인한다. 항목 SPID x ECID 0은 주 스레드를   
나타내며 항목 SPID x ECID > 0 은 같은 SPID에 대한 하위 스레드를 나타낸다.   
  
- Statement Type   
SELECT, INSERT, UPDATE 또는 DELETE 문 등이며, 스레드는 이에 대해 사용 권한을 갖는다.   
  
-Line #   
현재 명령문 배치에 있는 줄을 나열하는 데, 이 줄은 교착 상태가 발생할 때 실행된다.   
  
- Input Buf   
현재 배치에 있는 모든 명령문을 나열한다.   
  
- Mode   
스레드가 요청하고 허용하고 또는 대기하는 특정 리소스에 대한 잠금 유형을 보여준다.   
모드는 IS(Intent Shared), S(Shared), U(Update), IX(Intent exclusive), SIX(Shared with intent exclusive)   
및 X(Exclusive) 등이다.   
  
- RID   
잠금이 걸려 있거나 요구되는 테이블 안의 단일 행을 확인한다.   
  
- RID는 추적 플래그 1204에서 RID: db\_id:file\_id:page\_no:row\_no로 표시된다.   
예를 들면, RID: 1:1:1253:0입니다.   
  
- TAB   
잠금이 걸려 있거나 요구되는 테이블을 확인한다. TAB은 추적 플래그 1204에서 db\_id:object\_id로 표시됩니다.   
예를 들면 TAB:2:2009058193 이다.   
  
- KEY   
잠금이 걸려 있거나 요구되는 인덱스 안의 키 범위를 확인한다. KEY는 추적 플래그 1204에서   
KEY: db\_id:object\_id:index\_id로 나타난다. 예를 들면, KEY: 2:1977058079:1 이다.   
  
- PAG   
잠금이 걸려 있거나 요구되는 페이지 리소스를 확인한다. PAG는 추적 플래그 1204에서   
PAG: db\_id:file\_id:page\_no로 나타난다. 예를 들면, PAG: 7:1:168 이다.   
  
- EXT   
익스텐트 구조를 확인한다. EXT는 추적 플래그 1204에서 EXT: db\_id:file\_id:extent\_no로 나타난다.   
예를 들면, EXT: 7:1:9이다.   
  
- DB   
데이터베이스 잠금을 확인한다. DB는 추적 플래그 1204에서 다음 방법 중 하나로 나타난다.   
DB: db\_id   
DB: db\_id[BULK-OP-DB], 이것은 백업 데이터베이스가 갖는 데이터베이스 잠금을 확인한다.   
DB: db\_id[BULK-OP-LOG], 이것은 특정 데이터베이스에 대해 백업 로그가 갖는 잠금을 확인한다.   
  
- IND   
인덱스 리소스에서 만들어진 인덱스가 갖는 잠금을 확인한다. IND는 추적 플래그 1204에서   
다음 방법 중 하나로 나타난다.   
IND: db\_id:object\_id:index\_id   
IND: db\_id:object\_id:index\_id[INDEX\_ID], 이것은 인덱스 ID가 잠겼음을 나타난다.   
IND: db\_id:object\_id:index\_id[INDEX\_NAME], 이것은 인덱스 이름이 잠겼음을 나타난다.   
  
- APP   
응용 프로그램 리소스가 갖는 잠김을 확인한다. APP는 추적 플래그 1204에서 APP: lock\_resource로   
나타난다. 예를 들면, APP: Formf370f478 이다.   
  
SQL Server가 응용 프로그램 리소스를 교착 상태에서 처리하지 않으면, 해당 응용 프로그램   
리소스 소유자는 앞에서 설명한 오류 메시지를 받지 않는다.   
대신에 이 응용 프로그램은 sp\_getapplock 저장 프로시저가 해당 응용 프로그램 리소스에서 실행될 때   
"-3" 반환 코드를 받환받는다.   
  
- Victim Resource Owner   
교착 상태 주기를 끊기 위해 SQL Server가 처리하지 않는 진행 중인 스레드를 보여준다.   
선택된 스레드(SPID x ECID 0로 확인)와 기존의 모든 하위 스레드(SPID x ECID > 0로 확인)는 제거된다 .   
  
- Next Branch   
교착 상태 주기에 관련된 동일한 SPID에서 두 개 이상의 하위 스레드를 나타낸다.   
  
교착 상태가 병렬 처리와 관련이 있으면 여러 하위 스레드가 통신 버퍼에서 차단될 수 있으며,   
스레드 한 개는 다른 하위 스레드에 대해 대기 상태로 된다. 다른 모든 스레드가 교착 상태와 관련이   
있는 경우에만 교착 상태 상황이다. Next Branch는 대체 경로를 추적하는 교착 상태 주기를 나타낸다.   
  
  
/\* 참고   
일반적으로 SQL Server는 실행을 취소했을 때 가장 손해가 적은 트랜잭션을 실행하는 스레드를   
교착 상태 희생자로 선택한다. 또는 사용자가 SET 문을 사용하여 세션의 DEADLOCK\_PRIORITY를   
LOW로 설정할 수 있다. DEADLOCK\_PRIORITY 옵션은 교착 상태 상황에서 세션의 중요도를 판단하는   
방법을 제어한다. 세션이 LOW로 설정되어 있으면 교착 상태가 발생했을 때   
해당 세션이 희생자로 선택된다.   
\*/   
  
SET DEADLOCK\_PRIORITY   
이 명령은 교착 상태에 있을 때 세션이 반응하는 방법을 제어한다. 교착 상태는 두 프로세스에   
잠긴 데이터가 있고, 다른 프로세스가 그 잠금을 해제할 때까지 자신의 잠금을 해제하지 않을 때 발생한다.   
  
  
  
3) 교착상태 해결   
교착상태가 걸린 저장 프로시저나 일괄처리 내에서는 교착상태에 대한 오류처리를 할 수 없다.   
해당 처리가 종료된 후 따로 처리해 주어야 한다.   
  
<SESSION2>   
CREATE PROC UP\_LOCKTEST   
AS   
BEGIN TRAN   
UPDATE CUST   
SET CTEL = '032-333-9999'   
WHERE CID = 2   
--DBCC OPENTRAN   
--EXEC SP\_LOCK 52   
DELETE EMP   
WHERE EID = 1   
COMMIT TRAN   
GO   
  
IF @@ERROR = 1205   
BEGIN   
EXEC UP\_LOCKTEST   
SELECT 'OK'   
END   
  
  
4) 교착상태 최소화   
교착 상태를 완전히 피할 수는 없다. 그러나 교착 상태 수를 최소화할 수는 있다.   
교착 상태를 최소화하면 트랜잭션 처리량이 늘어나고 트랜잭션 수가 적어지기 때문에   
시스템 오버헤드가 줄어든다.   
  
1] 같은 순서로 개체에 액세스하게 한다.   
모든 동시 트랜잭션이 같은 순서로 개체에 액세스하면 교착 상태가 일어날 가능성이 줄어든다.   
예를 들어, 두 개의 동시 트랜잭션이 EMP 테이블에 대해 잠금을 얻은 다음 CUST 테이블에 대해   
잠금을 얻으면, 다른 트랜잭션이 완료될 때까지 한 트랜잭션이 EMP 테이블에서 BLOCKING 된다.   
첫 번째 트랜잭션이 COMMIT /ROLLBACK 하면 두 번째 트랜잭션은 계속 진행된다.   
그리고 교착 상태는 발생하지 않게 된다. 모든 데이터 수정에 대해 저장 프로시저를 사용하는 것은,   
개체 액세스 순서를 표준화할 수 있는 방법이 된다.   
  
2] 트랜잭션 중 사용자 상호 작용을 피한다.   
사용자 간섭 없이 실행 중인 일괄 처리의 속도는, 응용 프로그램에서 요청한 매개 변수에 대한   
프롬프트에 응답하는 등 사용자가 직접 쿼리에 응답하는 속도에 비해 매우 빠르므로   
사용자 상호 작용이 필요하도록 트랜잭션을 작성하지 않는 것이 좋다.   
예를 들어, 트랜잭션이 사용자 입력을 기다리고 있는데 사용자가 식사를 하러 가거나 퇴근한 경우   
사용자는 트랜잭션을 완료할 수 없다. 트랜잭션이 소유한 잠금은 트랜잭션이   
COMMIT /ROLLBACK 될 때만 해제되므로 이렇게 하면 시스템 처리량이 현저히 줄어들게 된다.   
교착 상태가 발생하지 않아도 같은 리소스에 액세스하는 다른 트랜잭션이 차단되므로 트랜잭션이   
완료되려면 기다려야 하기 때문이다.   
  
3] 트랜잭션을 하나의 일괄 처리로 짧게 유지한다.   
교착 상태는 보통 여러 개의 긴 트랜잭션이 같은 데이터베이스에서 동시에 실행될 때 발생한다.   
트랜잭션 실행 시간이 길수록 배타적 또는 업데이트 잠금 시간이 길어지고 다른 작업을 차단하여   
교착 상태가 발생할 가능성이 높아진다.   
그러나, 트랜잭션을 하나의 일괄 처리로 유지하면 트랜잭션 중 네트워크 왕복이 최소화되므로   
트랜잭션을 완료하고 잠금을 해제하는 데 걸리는 시간을 줄일 수 있다.   
  
4] 낮은 TRANSACTION\_ISOLATION 레벨을 사용한다.   
트랜잭션을 더 낮은 격리 수준에서 실행할 수 있는지 확인해 본다. READ UNCOMMITTED 를 구현하면   
첫 번째 트랜잭션이 완료될 때까지 기다리지 않고 이전에 읽은(수정하지 않은) 데이터를 읽을 수 있다.   
READ UNCOMMITTED 등 낮은 잠금 수준을 사용하면 SERIALIZABLE 등의 높은 잠금 수준보다   
짧은 기간 동안 공유 잠금을 보유하므로 잠금 경쟁률이 줄어든다.   
  
5] 바운드 연결을 사용한다.   
바운드 연결을 사용하면 같은 응용 프로그램에서 열고 있는 둘 이상의 연결을 함께 사용할 수 있다.   
두 번째 연결에서 얻은 잠금은 주 연결에서 얻은 것처럼 보유되며 그 반대의 경우도 마찬가지로 처리된다.   
따라서 서로를 BLOCKING 하지 않는다.

출처:<http://blog.boyo.kr/entry/MSSQL-LOCK-%EC%9D%B4%EB%9E%80>

출처: <http://battleracoon.tistory.com/4> [개발하는 너구리]

**인덱스와 잠금**

InnoDB의 잠금과 인덱스는 상당히 중요한 연관 관계가 있다.

InnoDB의 잠금은 레코드를 잠그는 것이 아니라 인덱스를 잠그는 방식으로 처리된다. *즉, 변경해야 할 레코드를*

*찾기 위해 검색한 인덱스의 레코드를 모두 잠가야 한다.*

*만약 테이블에 인덱스가 없다면 테이블을 풀 스캔하면서 작업을 하는데 이 과정에서 테이블에 있는 모든 레코드를*

*잠그게 된다. 이것의 MySQL의 방식이며, MySQL의 InnoDB에서 인덱스 설계가 중요한 이유 또한 이 때문이다.*

출처: [http://imdsoho.tistory.com/entry/트랜잭션과-잠금](http://imdsoho.tistory.com/entry/%ED%8A%B8%EB%9E%9C%EC%9E%AD%EC%85%98%EA%B3%BC-%EC%9E%A0%EA%B8%88) [삶은 예배다.]

데드락 관련 상세한 정리 : <https://www.letmecompile.com/mysql-innodb-lock-deadlock/>

데드락 관련 자료들 모아놓음 : <https://m.blog.naver.com/PostView.nhn?blogId=parkjy76&logNo=220600231018&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F>

http://www.mysqlkorea.com/sub.html?mcode=manual&scode=01&m\_no=21889&cat1=14&cat2=422&cat3=444&lang=k