# 12. 트랜잭션과 잠금



여러 개의 데이터 수정이 하나의 단위로 수행되어야 하는 경우가 많다. 그래서 이 중 하나라도 문제가 발생한다면 모든 처리가 다 취소 되어야 한다. 모두 다 처리가 되던지 아니면 아예 하나도 처리가 안되든지...이것이 트랜잭션이다.

# 사랑 차 만드는 법

사랑 차 준비물

1. 성냄과 불평은 뿌리를 잘라내고 잘게 다진다.
2. 교만과 자존심은 속을 빼낸 후 깨끗이 씻어 말린다.
3. 짜증은 껍질을 벗기고 반으로 토막을 낸 후에 넓은 맘으로 절여 둔다.

사랑 차 끓이는 방법

1. 주전자에 실망과 미움을 한 컵씩 붓고, 씨를 잘 빼낸 다음 불만을 넣고 푹 끓인다.
2. 미리 준비한 재료를 인내와 기도를 첨가하여 재료가 다 녹고 쓴맛이 없어지기까지 충분히 달인다.
3. 기쁨과 감사로 잘 젓고, 미소를 몇 개 띄운 후 깨끗한 믿음으로 잔에 부어서 따뜻하게 마신다.

# 트랜잭션과 잠금
## (Transactions and Locks)

- 트랜잭션(Transactions)
  - 트랜잭션은 여러 데이터 수정이 하나의 단위로 처리되게 함
  - 트랜잭션은 잠금을 사용하여 완료되지 않은 트랜잭션의 데이터를 다른 사용자가 변경 또는 읽을 수 없게 함
  - 다중 사용자 시스템을 위한 온라인 트랜잭션 처리(OLTP)에는 잠금이 필요함
  - 다중 트랜잭션을 관리 할 수 있도록 SQL Server는 트랜잭션 처리를 지원함
- 잠금(Locks)
  - 잠금은 업데이트 충돌을 방지함
  - 특정 사용자는 다른 사용자가 변경 중인 데이터를 읽거나 수정할 수 없음

# 트랜잭션 제어
## (Controlling Transactions)

- **트랜잭션 시작**
  - 명시적 트랜잭션(Explicit Transactions)
    - BEGIN TRANSACTION으로 트랜잭션 시작
  - 자동 커밋 트랜잭션(Auto Commit Transactions)
    - MS SQL 서버의 기본 모드
    - 각 Transact-SQL문은 완료 시 자동으로 커밋 됨
  - 암시적 트랜잭션(Implicit Transactions)
    - SET IMPLICIT_TRANSACTION ON으로 설정
- **트랜잭션 완료**
  - COMMIT
    - 모든 트랜잭션에 의한 수정이 데이터베이스에 영구적으로 적용됨
  - ROLLBACK
    - 데이터를 트랜잭션 시작 전 상태로 되돌려 놓음

# 암시적 트랜잭션 옵션
## (Implicit Transactions Option)

- 암시적 트랜잭션 모드가 설정되면 특정 문 수행 시 자동으로 트랜잭션이 시작됨
- 중첩된 트랜잭션은 허용되지 않음
- COMMIT 또는 ROLLBACK TRANSACTION을 통해 트랜잭션이 명시적으로 커밋되거나 롤백 되어야 함
- 이 옵션은 기본적으로 OFF 로 설정되어 있음

```
SET IMPLICIT_TRANSACTIONS ON
```

# DEMO



트랜잭션의 사용 예를 살펴 봄

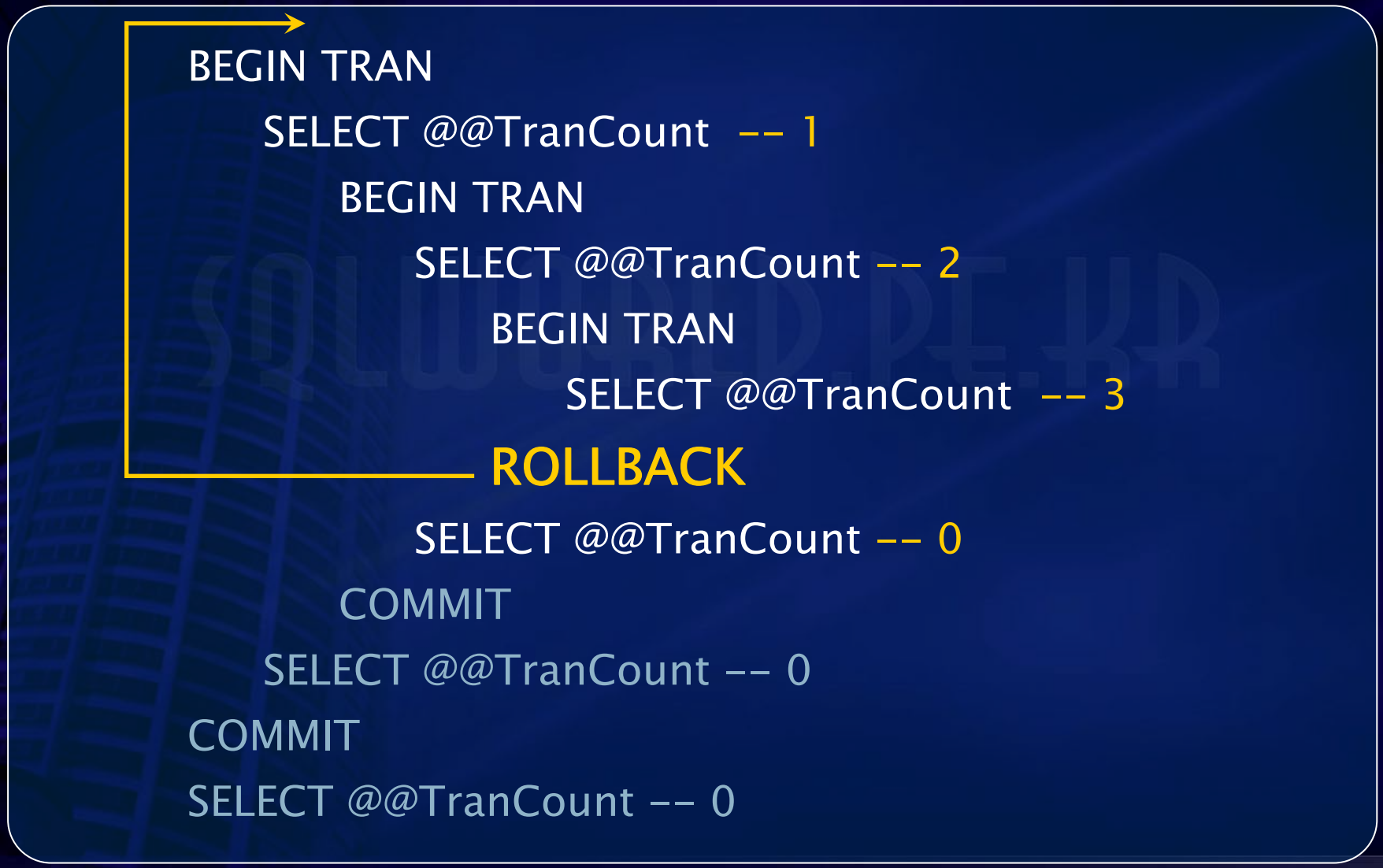
# 중첩 트랜잭션 (1)

(Nested Transactions)

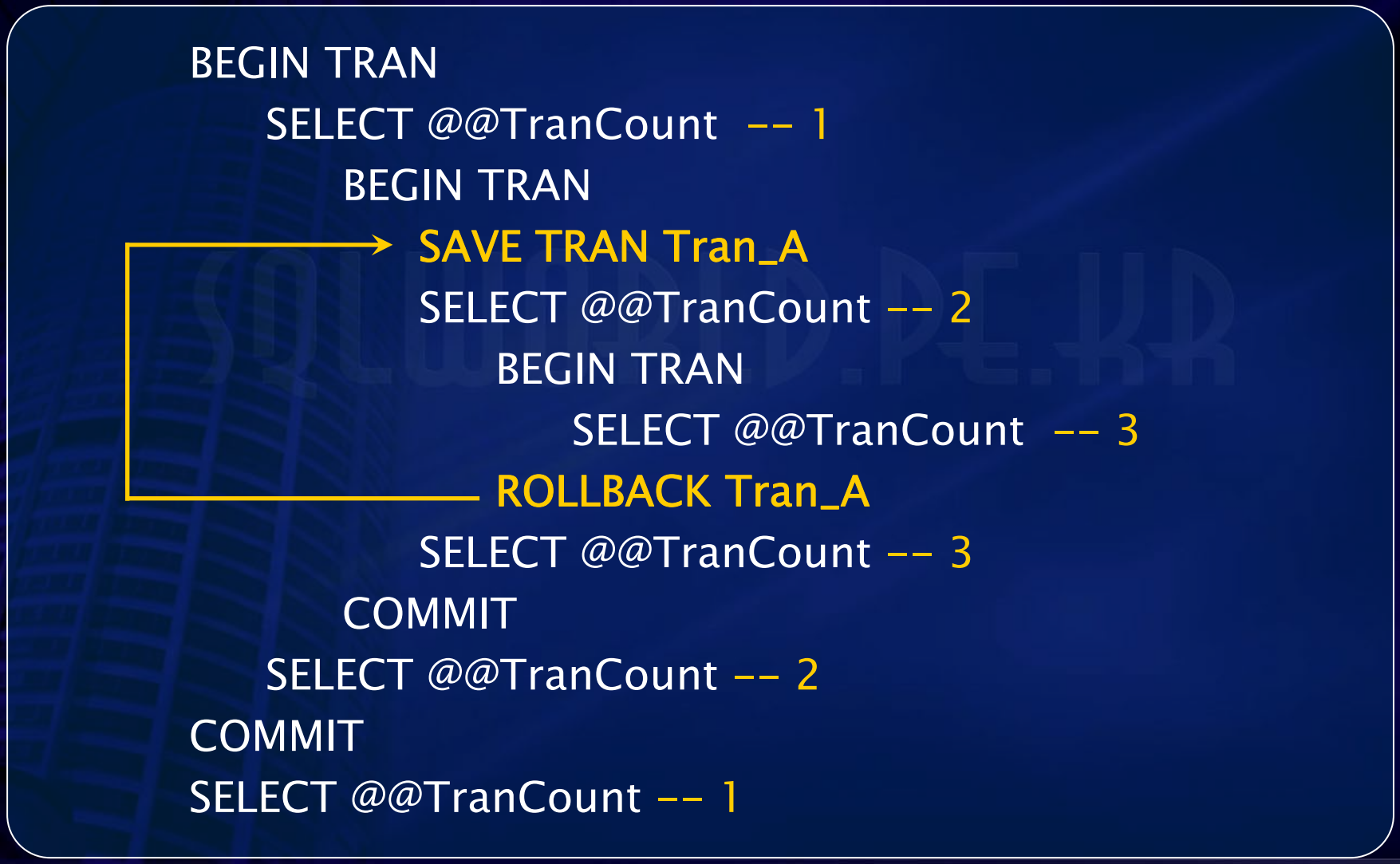
```
BEGIN TRAN
    SELECT @@TranCount  -- 1
        BEGIN TRAN
            SELECT @@TranCount -- 2
                BEGIN TRAN
                    SELECT @@TranCount  -- 3
                COMMIT
            SELECT @@TranCount -- 2
        COMMIT
    SELECT @@TranCount -- 1
COMMIT
SELECT @@TranCount -- 0
```

# 트랜잭션 제어 (2)

## (Nested Transactions)



# 포인트 설정
(Save Point)



# DEMO



트랜잭션의 중첩과 Save Point에 대해 살펴 봄

# 트랜잭션 지원을 위한 속성

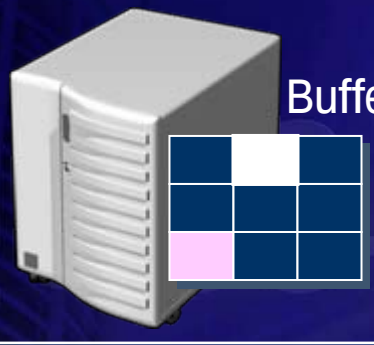
(ACID)

| 속성 | 설명 |
| --- | --- |
| 원자성 (Atomicity) | 트랜잭션의 모든 연산들이 데이터베이스에 모두 적용되던지 또는 모두 적용되지 않아야 한다. |
| 일관성 (Consistency) | 데이터에 대한 무결성이 유지되기 위해서는 데이터베이스 내의 모든 규칙이 트랜잭션에 적용되어야 한다. 즉 규칙에 위반되는 트랜잭션은 무결성을 위해 취소되어야만 데이터베이스에 일관성을 유지할 수 있게 된다. |
| 고립성 (Isolation) | 현재의 트랜잭션에 의해 변경되고 있는 데이터 영역은 다른 사용자의 트랜잭션 영역으로부터 분리되어야 한다. |
| 영속성 (Durability) | 트랜잭션이 정상적으로 종료된 후에는 시스템 오류가 발생하더라도 데이터를 안정적으로 저장시켜야 한다. |

# 트랜잭션 처리

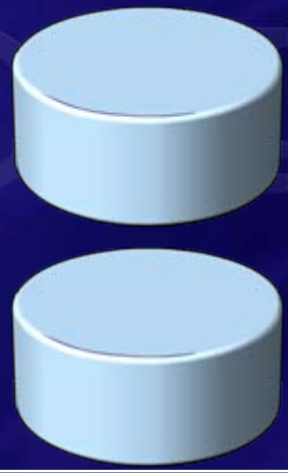
## (How the Transaction Log Works)

1 사용자가 데이터베이스 변경작업을 수행함




2 관련된 데이터 페이지가 Cache에 있는지 확인하여 Cache에 있다면 그 내용을 수정하고, Cache에 없다면 Disk에서 Cache로 가져온 후 수정

3 변경된 내용을 Disk에 있는 데이터베이스의 트랜잭션 로그 파일에 기록함



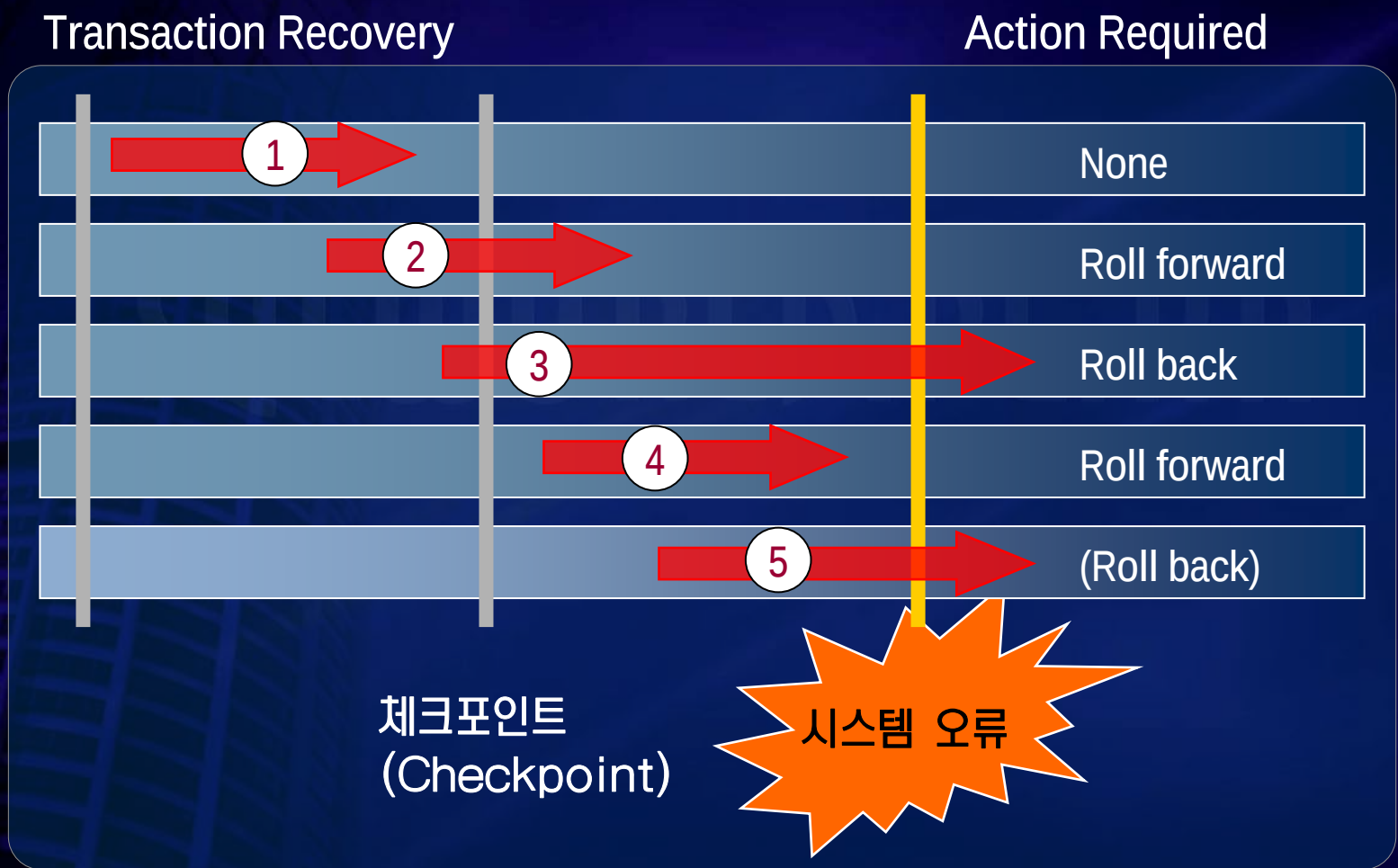

4 Checkpoint라 불리는 프로세스가 변경된 페이지(Dirty Pages)를 Cache로부터 Disk에 있는 데이터베이스의 데이터 부분으로 밀어냄

# 롤 포워드와 롤 백
## (Roll Forward and Roll Back)

- 트랜잭션 로그로부터 정보를 얻어 처리함
- SQL Server 시작 시 처리 프로세스
  - 모든 데이터베이스에 대해서 트랜잭션 로그에 기록된 가장 최근의 Checkpoint 시점을 확인
  - 이 Checkpoint 이후에 Commit된 액션에 대해 롤 포워드 됨 (Redo Phase)
  - 트랜잭션 로그에 COMMIT TRAN 기록이 없는 모든 오픈 된 트랜잭션은 롤 백 됨 (Undo Phase)
  - 롤 포워드와 롤 백이 모두 수행 된 후 데이터베이스 접근이 가능해짐
  - 단 SQL Server 2005는 Redo 이후 접근 가능

# 트랜잭션 복구 및 검사점

## (Transaction Recovery Works)



# 트랜잭션 사용을 위한 고려사항
(Considerations for Using Transactions)

- 트랜잭션을 짧게 유지
  - WHILE문과 같은 특정 Transact-SQL문 사용 시 주의
  - 트랜잭션 중 사용자 입력을 요구하지 않음
  - INSERT, UPDATE, DELETE 문이 최소한의 행 수에 영향을 주도록 함
  - 데이터 검색하는 동안은 가능하면 트랜잭션을 열지 않음
  - 트랜잭션 중 가능한 최소 양의 데이터 액세스
- 중첩 트랜잭션(Nesting Transactions)
  - 트랜잭션은 중첩 될 수는 있지만 권하지는 않음
  - @@trancount를 이용하여 중첩의 깊이를 확인함

# 잠금 모드 (1)
(Lock Mode)

- 공유 잠금(Shared Lock) : S
  - 데이터를 읽을 때 자동으로 공유 잠금을 얻음
  - 여러 프로세스가 동일한 데이터에 대한 공유 잠금 유지 가능
  - 공유 잠금이 있는 데이터에서 단독 잠금을 얻을 수 없음
  - 일반적으로 데이터를 읽자마자 공유 잠금 해제됨
  - 쿼리 힌트, 트랜잭션 격리 수준을 이용해 공유 잠금 해제를 제어 할 수 있음
- 단독 잠금(Exclusive Lock) : X
  - 데이터가 삽입, 업데이트, 삭제에 의해 변경 될 때 단독 잠금을 얻음
  - 한 번에 한 프로세스만이 단독 잠금을 유지 할 수 있음
  - 트랜잭션이 커밋되거나 롤백될 때까지 다른 프로세스가 변경된 데이터를 사용할 수 없음
  - 힌트를 사용함으로써 다른 프로세스들이 단독 잠금이 유지된 데이터를 읽을 수 있음

# 잠금 모드 (2)
(Lock Mode)

- 업데이트 잠금(Update Lock) : U
  - 변경될 리소스를 찾기 위해 테이블을 검색할 필요가 있을 때 업데이트 잠금이 얻어짐
  - 데이터를 읽은 후로 데이터가 변경되지 않았다는 것을 확신 하면서 나중에 데이터를 변경 할 수 있게 함
  - 나중에 단독 잠금을 요청 할 수 있게 하는 관문 역할
  - 한 프로세스만이 업데이트 잠금을 유지 할 수 있음
  - 특정 프로세스가 업데이트 잠금을 유지하는 한 다른 프로세스가 이 자원에 대해 업데이트 잠금이나 단독 잠금을 얻을 수 없음
- 내재된 잠금(Intent Lock) : IS, IU, IX
  - 리소스의 구성 요소가 이미 잠겨 있음을 알려주는 잠금
  - 내재된 공유 잠금
  - 내재된 단독 잠금
  - 내재된 업데이트 잠금

# 잠금 모드 (3)
(Lock Mode)

- 스키마 안정성 잠금(Schema Stability Lock) : Sch-S
  - 쿼리가 컴파일 될 때 다른 프로세스들이 스키마 수정 잠금을 얻지 못하도록 함
- 스키마 수정 잠금(Schema Modification Lock) : Sch-M
  - 개체에 대해 DDL문이 수행 될 때 얻어 짐
- 대량 업데이트 잠금(Bulk Update Lock) : BU
  - 다음 중 한가지가 설정된 상태에서 대량 복사 작업이 수행되는 경우 얻어 짐
    - TABLOCK 힌트 설정
    - 'table lock on bulk load' 테이블 옵션이 설정된 경우
- 키 범위 잠금(Key-range Lock) : Range...
  - Serializable 격리 수준에서 데이터 범위를 잠그려고 할 때 얻어 짐

# DEMO



잠금 모드의 실제 발생 사례를 살펴 보고 잠금 정보가
표시하는 내용의 의미를 파악 함

# 잠금 호환성
## (Lock Compatibility)

| 요청된 모드 (Requested Mode) | 기존에 허용된 모드 (Granted Mode) | | | | |
|---|---|---|---|---|---|
| | **IS** | **S** | **U** | **IX** | **X** |
| Intent shared (IS) | O | O | O | O | X |
| Shared (S) | O | O | O | X | X |
| Update (U) | O | O | X | X | X |
| Intent exclusive (IX) | O | X | X | O | X |
| Exclusive (X) | O | X | X | X | X |

# 잠금으로 방지되는 동시성 문제
## (What Concurrency Problems Are Prevented by Locks?)

- 손실된 업데이트(Lost Update)
  - 특정 트랜잭션이 다른 트랜잭션의 변경 내용을 덮어 쓸 경우 업데이트가 손실 됨
- 커밋되지 않은 데이터 읽기(Dirty Read)
  - 특정 트랜잭션이 다른 트랜잭션의 커밋되지 않은 데이터를 읽은 경우 발생
- 일관성 없는 분석(Inconsistent Analysis)
  - 특정 트랜잭션이 같은 행을 두 번 이상 읽을 경우 읽기 사이에 다른 트랜잭션이 해당 행을 수정하면 일관성 없는 분석 발생
- 팬텀 읽기(Phantom Reads)
  - 트랜잭션이 서로 격리되지 않을 경우 발생

# 트랜잭션 격리 수준
## (Transaction Isolation Level)

- READ UNCOMMITTED
  - NOLOCK을 설정하는 것과 같아 데이터를 수정하는 다른 세션과 충돌 없음
  - Dirty Read를 포함해 앞 페이지에서 나열한 동시성 문제가 발생함
  - 다른 격리 수준에 비해 일관성을 가장 떨어지지만 동시성은 가장 뛰어남
- READ COMMITTED (기본값)
  - 데이터를 읽을 때는 공유 잠금이 유지되도록 해서 커밋되지 않은 데이터 읽기가 이루어지지 않도록 지정함
  - 즉 Dirty Read 가 발생하지 않음
  - 트랜잭션이 끝나기 전에 데이터가 변경되어 반복하지 않는 읽기 또는 팬텀 데이터가 만들어질 수 있음
- REPEATABLE READ
  - 공유 잠금을 트랜잭션이 종료 할 때까지 유지하여 다른 사용자가 데이터를 업데이트할 수 없도록 함
  - 다른 사용자가 데이터 집합에 새 허위 행을 삽입해 현재 트랜잭션의 이후 읽기에 포함될 수 있음, 즉 팬텀 데이터 존재

# 트랜잭션 격리 수준
(Transaction Isolation Level)

- SERIALIZABLE
  - 데이터 집합에 범위 잠금을 배치함
  - 트랜잭션이 완료될 때까지 다른 사용자가 행을 업데이트하거나 데이터 집합에 삽입할 수 없도록 함
  - 팬텀 데이터가 발생하지 않음
- SQL Server 2005에 추가된 격리 수준
  - Snapshot
  - Read Committed Snapshot

# DEMO



트랜잭션 격리 수준의 네 가지 예를 살펴 봄으로써
각각의 특성을 파악함

# 격리 수준 요약

## (Summary of Isolation Levels)

| 격리 수준 | Dirty Reads | Lost Updates | Nonrepeatable Reads | Phantoms |
|---|---|---|---|---|
| Read Uncommitted | O | O | O | O |
| Read Committed | X | O | O | O |
| Repeatable Read | X | X | X | O |
| Serializable | X | X | X | X |

# 교착 상태
(Deadlocks)

- 교착 상태의 희생자(Victim)

  메시지 1205, 수준 13, 상태 51, 줄 2
  트랜잭션(프로세스 ID 54)이 잠금 리소스에서 다른 로세스와의 교착 상태가 발생하여 실행이 중지되었습니다. 트랜잭션을 다시 실행하십시오.

- SET DEADLOCK_PRIOROTY
  - LOW
  - NORMAL (default)
  - HIGH
- 인덱스가 없는 경우 교착 상태 발생 가능
- 하나의 테이블에서도 교착 상태 발생 가능

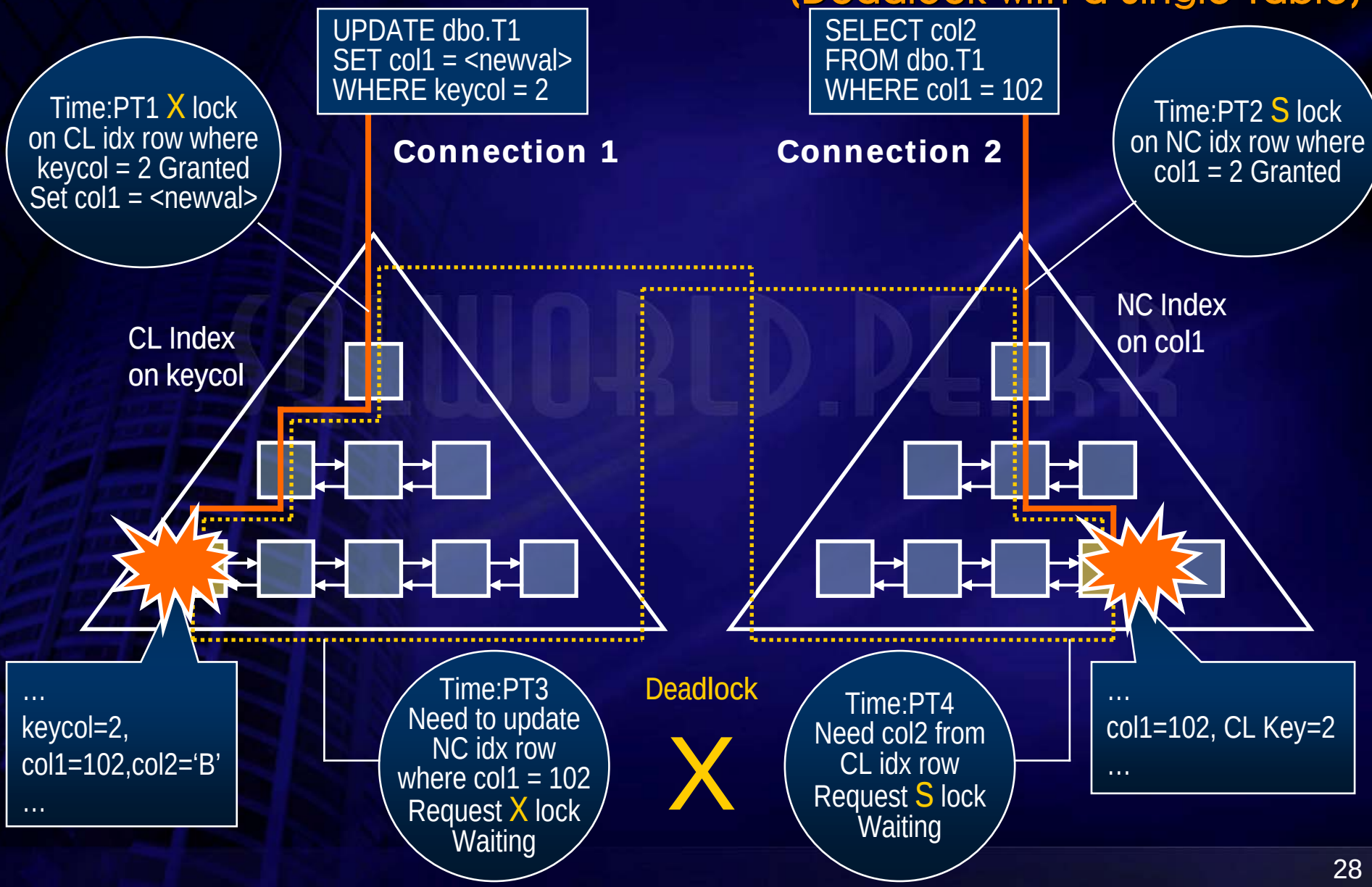
# DEMO



교착 상태가 발생할 수 있는 경우에 대해 살펴 봄

# 하나의 테이블에서 교착 상태

## (Deadlock with a Single Table)



# 잠금 정보 표시
(View Locking Information)

- EM의 현재 동작 창(Current Activity Window)
- sp_lock 시스템 저장 프로시저 사용
- SQL 프로필러(SQL Profiler) 사용
- 윈도우의 시스템 모니터(System Monitor) 사용
- syslockinfo와 같은 시스템 테이블 이용

# 정리