DBMS는 어떻게 트랜잭션의 ACID 원칙을 준수할까?

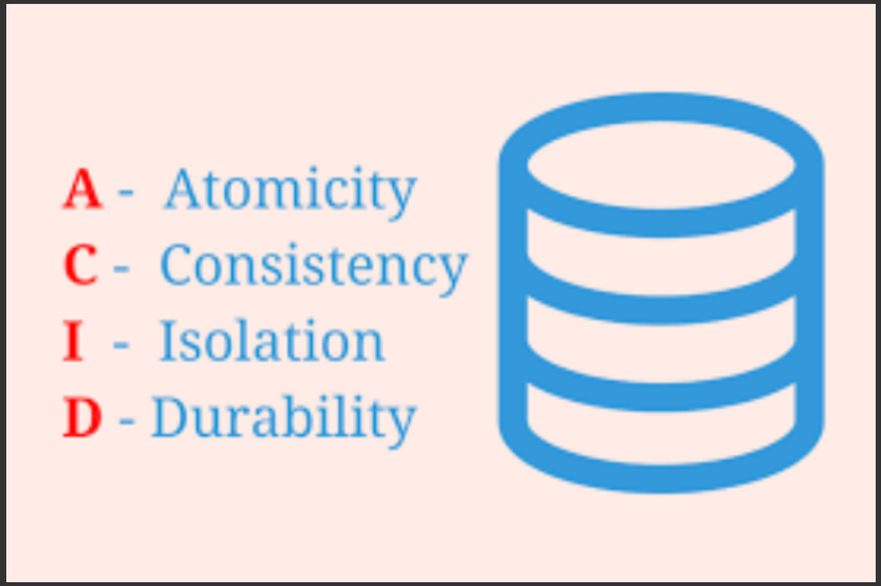
발표자: 한지원

Content

- 트랜잭션 고립화 수준의 정의
- 트랜잭션 고립화 수준의 종류
- 트랜잭션 고립화 수준에 따른 이상 현상
- MySQL vs Oracle의 트랜잭션 고립화 수준 구현
- 다중 버전 동시성 제어
- Undo 영역과 Redo 영역

선수 지식

트랜잭션의 ACID 원칙



DBMS는 어떻게 일관성, 고립성을 보장할까?

Transaction Isolation Level

트랜잭션 수준 읽기 일관성: 트랜잭션 시작 지점을 기준으로 일관성 있도록 데이터를 읽는 것

트랜잭션 고립화 수준: 트랜잭션끼리 어느 정도의 데이터 일관성을 보장할 것인가를 수준 별로 정해 놓은 것

Transaction Isolation Level

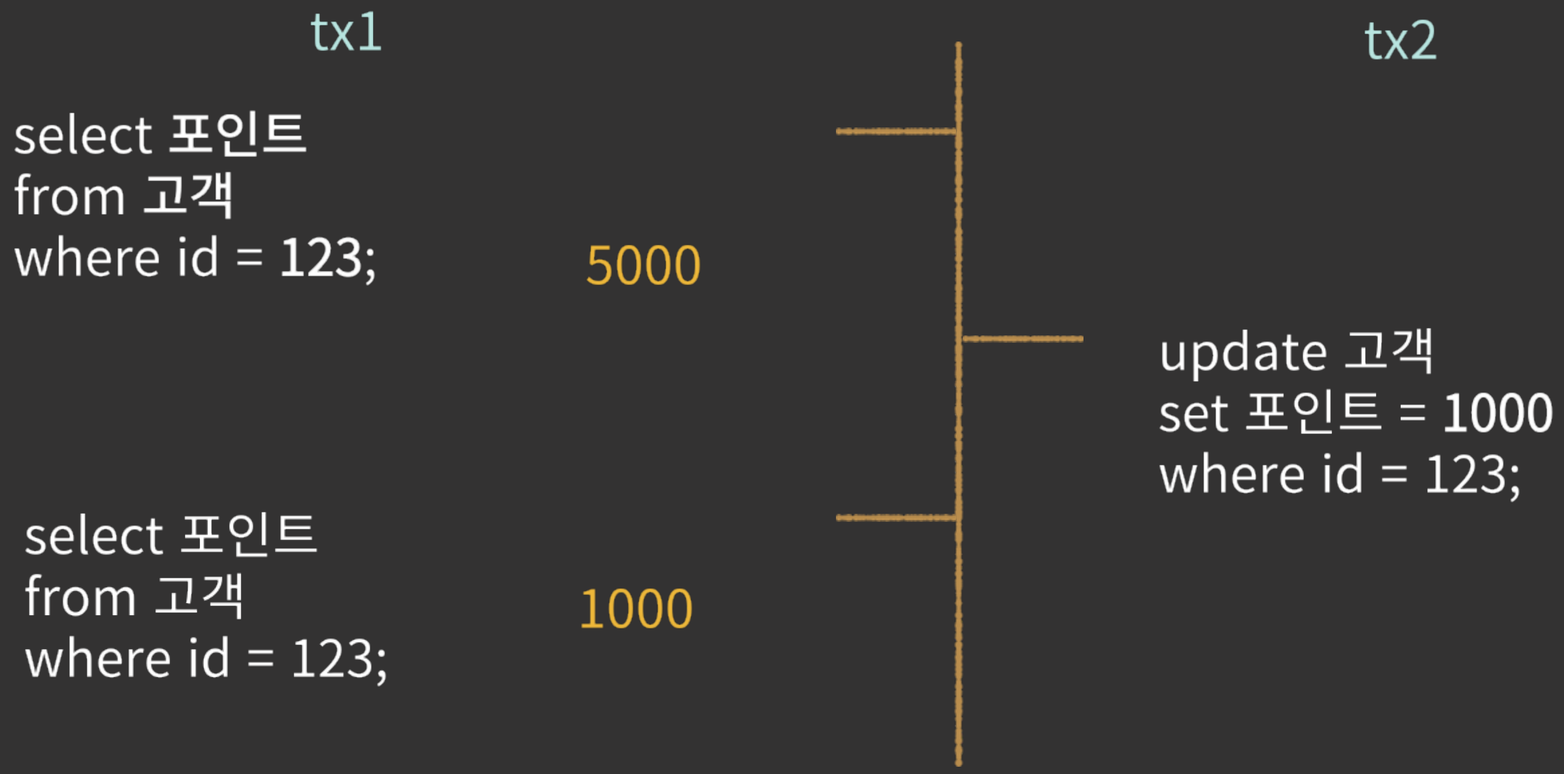
동시성

높은 격리성 수준 및 데이터의 일관성

Transaction 이상 현상

1. Dirty Read

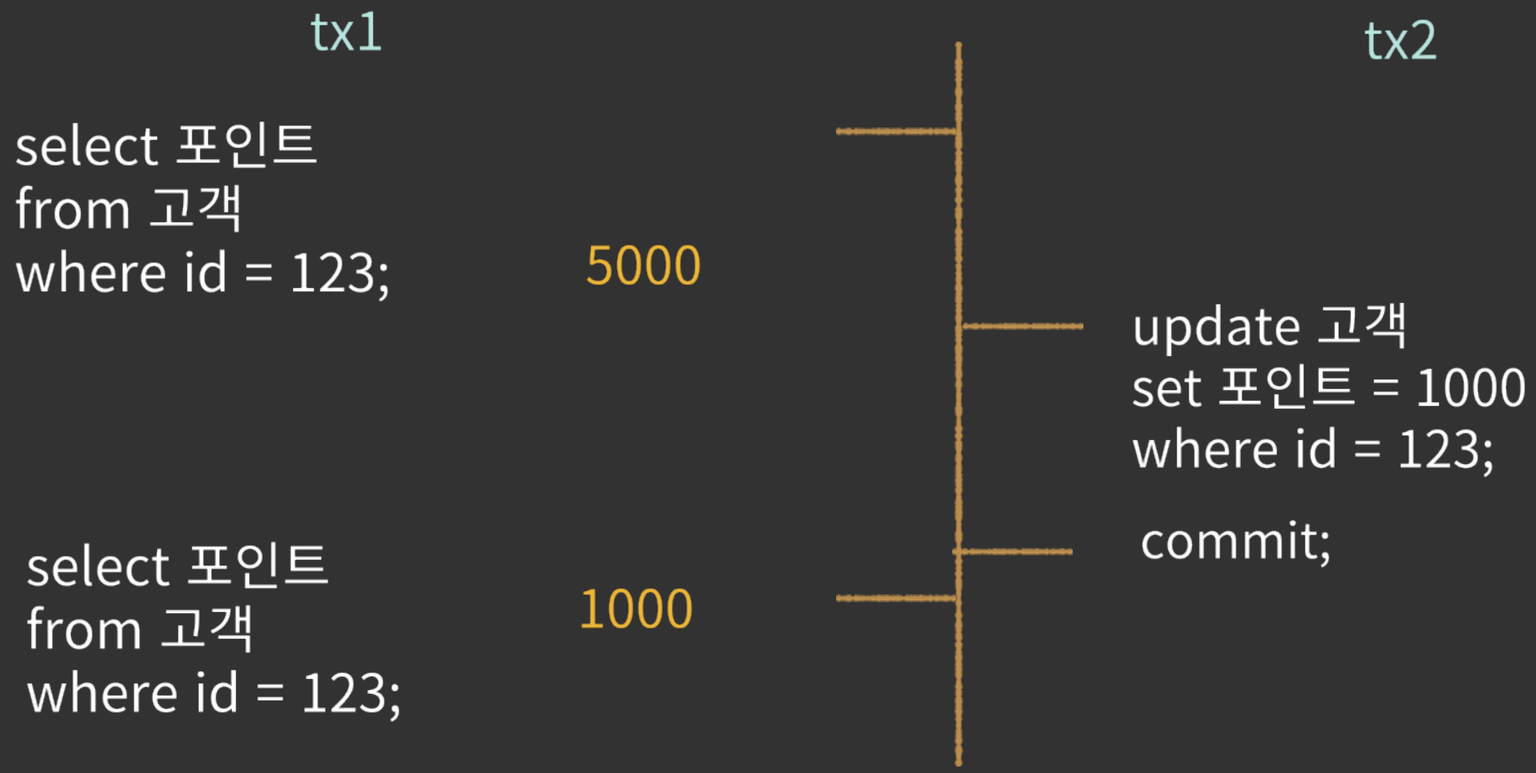
: 아직 커밋되지 않은 수정중인 데이터를 다른 트랜잭션에서 읽을 수 있도록 허용



Transaction 이상 현상

2. Non-Repeatable Read

: 같은 쿼리를 중복해서 수행할 경우, 그 사이에 다른 트랜잭션의 값 수정 및 삭제로 인해 두 쿼리의 결과가 상이하게 나타나는 비일관성 발생



Transaction 이상 현상

3. Phantom Read

: **일정 범위**의 레코드를 조회하는 쿼리를 중복해서 수행할 경우, 그 사이에 다른 트랜잭션의 삽입 연산으로 인해 이전에는 없던 **유령 레코드**가 나타나는 현상

tx1

```
insert into 지역별고객
select area, count(*)
from 고객
group by area;

insert into 연령대별고객
select age, count(*)
from 고객
group by age;

commit;
```

tx2

```
insert into 고객(id,name,area, age)
values (20, :'한지원', '서울', 20);

commit;
```

Transaction Isolation Level

Level 0. Read Uncommitted

Level 1. Read Committed

Level 2. Reapeatable Read

Level 3. Serializable

Transaction Isolation Level

Concurrency

Consistency

	Dirty Read	Non-Repeatable Read	Phantom Read
Read Uncommitted	발생	발생	발생
Read Committed	발생 X	발생	발생
Reapeatable Read	발생 X	발생 X	발생
Serializable	발생 X	발생 X	발생 X

DBMS 별 어떤 고립 수준을 선택해야 할까?

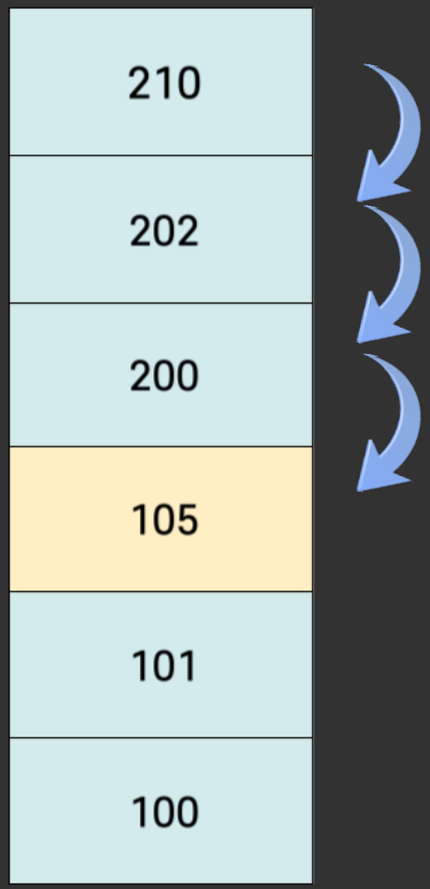
MySQL vs Oracle

	MySQL	Oracle
default isolation level	Repeatable Read	Read Committed
지원되는 isolation level	Read Uncommitted, Read Committed, Repeatable Read, Serializable	Read Committed, Serializable, (Repeatable Read는 select for update문을 통해 제공)
Strage Engine	MyISAM,InnoDB, Aria 등	단일 Stroage Engine

DBMS는 어떤 방법으로
트랜잭션 고립수준을 제공할까?

다중 버전 동시성 제어(MVCC)

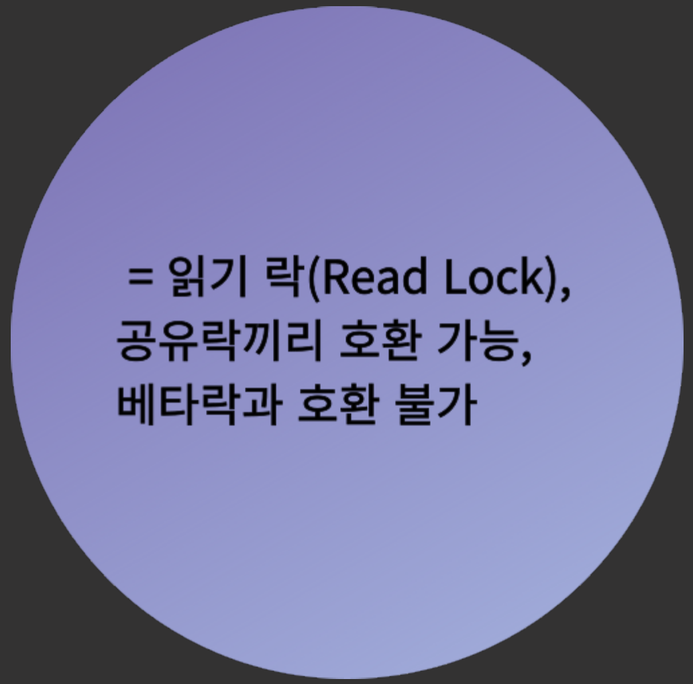
: 레코드의 잠금 없이 포인트-인-타임 일관성(point-in-time consistent) 뷰를 제공하여 트랜잭션 내에 일관성 있는 읽기 제공한다.
일반적으로 타임스탬프나 트랜잭션 ID를 사용하여 읽을 DB의 상태를 결정한다.



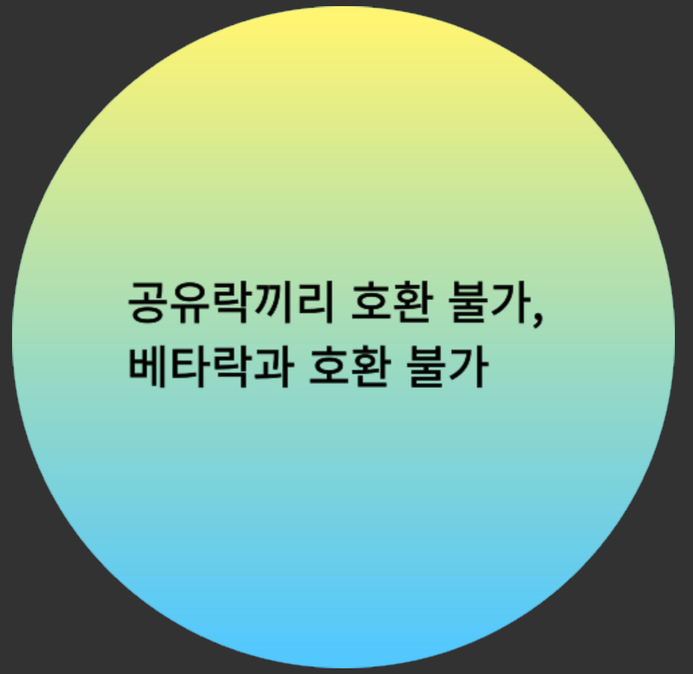
Undo Segment

Lock

공유 락(Shared Lock)



베타 락(Exclusive Lock)



Isolation Level in MySQL, Oracle

	MySQL	Oracle
Shared Lock(읽기 락)	사용 O	공유락 사용 X, 읽기와 쓰기 작업이 관여하지 X
Read Uncommitted	어떠한 Locking도 사용하지 않음	X
Read Committed	트랜잭션 내 select문 마다 새로운 스냅샷을 읽음 SELECT with FOR UPDATE or FOR SHARE, UPDATE, DELETE 수행 시 인덱스 레코드에만lock, gap lock 사용X Phantom Read 발생O	기본적으로 커밋된 데이터만 읽는다.
Repeatable Read	트랜잭션 내 모든 select문이 동일한 스냅샷을 읽음 unique index 검색 -> 해당 인덱스 레코드에만lock 설정 그 외 -> gap lock or next-keylock으로 인덱스 스캔 범위에 락 사용. -> Phantom read 발생 X	X, select for update를 통해 사용 가능
Serializable	Repeatable Read와 유사. autocommit disabled로 설정 시, InnoDB에서는 모든 plain select를 select for share로 변환	트랜잭션 시점의 데이터를 읽는다.

DBMS는 어떻게 영속성과 원자성을 보장할까?

트랜잭션의 Durabiltiy을 위협하는 요인
= 메모리 캐시

Redo & Undo

Redo

모든 변경사항을 기록하는 것.

Undo

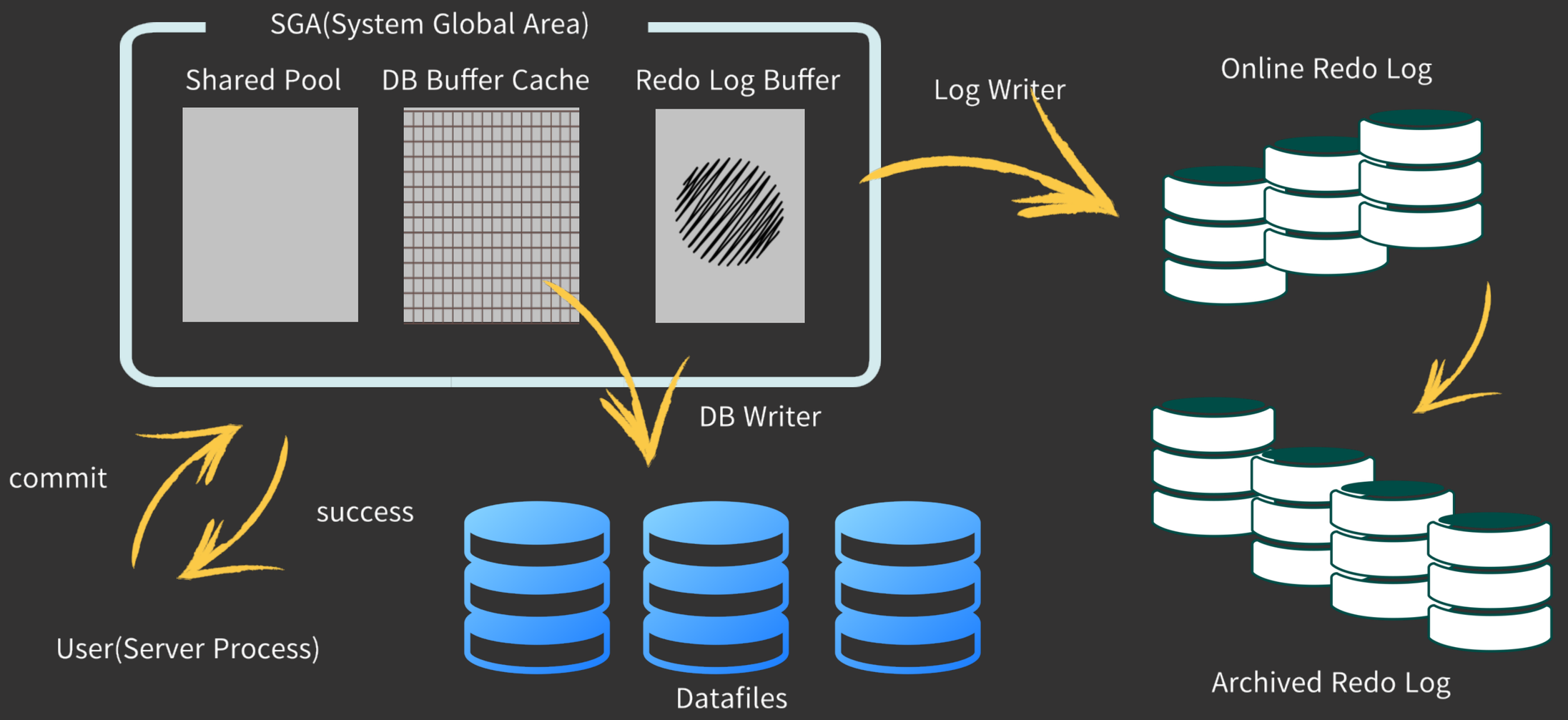
변경사항을 되돌리기 위한 트랜잭션 정보를 기록하는 것

Redo Log

1. Database Recovery	물리적으로 디스크가 깨지는 등의 Media Fail 발생 시 데이터베이스 복구에 Archived Redo Log 사용
2. Cache Recovery	휘발성인 버퍼 캐시를 복구하는 데 사용. Online Redo Log에 저장된 기록을 통해 Instance Crash 마지막 체크포인트 이후부터 사고 발생 직전까지의 트랜잭션들을 재현. **트랜잭션 커밋여부를 불문하고 버퍼 캐시를 시스템 셧다운 이전 상태로 되돌리는 것**
3. Fast Commit	append 방식으로 빠른 속도로 Redo Log에 저장 후, 실제로 데이터 파일과의 동기화는 DBWR을 이용해 추후 배치 방식으로 처리하는 방법

Redo

Write Ahead Logging "로그먼저 미리 쓰기"

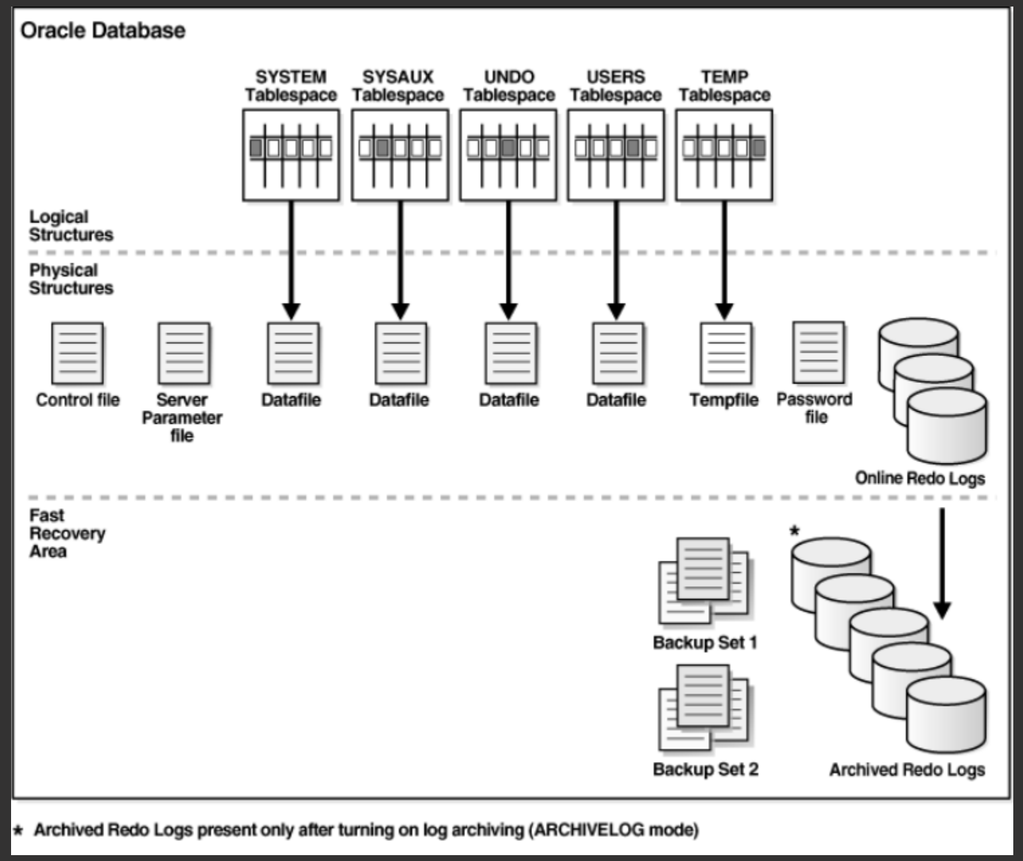


Log First at Commit "커밋 시점에 데이터 파일보다 로그 먼저"

Undo(=Rollback)

1. Transaction Rollback	트랜잭션에 의한 변경사항을 **최종 커밋하지 않고 롤백**할 때 사용
2. Transaction Recovery	시스템 셧다운 발생 시 **아직 커밋되지 않았던 트랜잭션을 모두 롤백**하는 방법
3. Read Consistency(Oracle)	Oracle에서 읽기 일관성을 구현하는 방법

Undo(=Rollback)



변경되는 컬럼의 before image(update)

Undo Header

추가된 레코드의 rowid (insert)

삭제되는 로우에 대한 before image (delete)

업데이트 컬럼 before image (update)

transatcion ID, commit SCN, Last UBA, 트랜잭션 상태 정보

DBMS 장애 Recovery Process

tx1

A 계좌에서 200원을 인출 A: 800

commit;

tx2

B 계좌에 200원을 입금 B: 2,200

장애 발생!

DB Buffer Cache

계좌	잔액
A	1000 -200
B	2000 +200

Datafile

계좌	잔액
A	1000
B	2000

DBMS 장애 Recovery Process

tx1

A 계좌에서 200원을 인출 A: 800

commit;

tx2

B 계좌에 200원을 입금 B: 2,200

장애 발생!

DB Buffer Cache

계좌	잔액
A	
B	

Datafile

계좌	잔액
A	1000
B	2000

DBMS 장애 Recovery Process

tx1

A 계좌에서 200원을 인출 A: 800

commit;

tx2

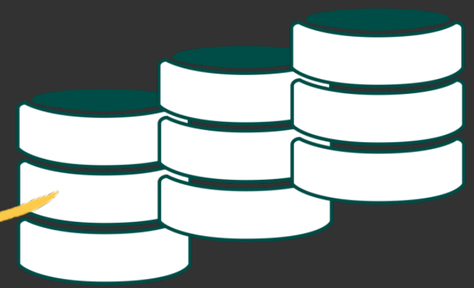
B 계좌에 200원을 입금 B: 2,200

장애 발생!

DB Buffer Cache

계좌	잔액
A	1000 -200
B	2000 +200

Online Redo Log



DBMS 장애 Recovery Process

tx1

A 계좌에서 200원을 인출 A: 800

commit;

tx2

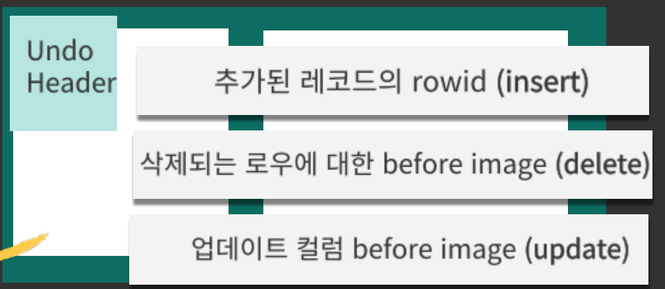
B 계좌에 200원을 입금 B: 2,200

장애 발생!

DB Buffer Cache

계좌	잔액
A	800
B	2000 +200

Undo Segment



Review

- 트랜잭션 고립화 수준: Read Uncommitted, Read Committed, Repeatable Read, Serializable
- 트랜잭션 고립화 수준에 따른 이상 현상: Dirty Read, Non-repeatable Read, Phantom Read
- 트랜잭션 고립화 수준과 동시성의 trade-off 관계
- 다중 버전 동시성 제어과 Lock
- Redo 와 Undo를 통한 트랜잭션 영속성 보장

생각해볼 문제

select for update를 써야할 경우는 언제일까?

Reference

- 오라클 성능 고도화 원리와 해법 - 조시형
- https://dev.mysql.com/doc/refman/9.0/en/glossary.html#glos_record_lock
- https://docs.oracle.com/en/database/other-databases/timesten/22.1/introduction/transaction-isolation.html#GUID-117644E7-8EC6-4332-86AF-0549B6C5C506

The End ...