

# Buffer States

Oracle 19c Database Concepts 기반 학습 문서

백엔드 개발자 관점의 아키텍처 심층 분석

## 1. 개요: 버퍼 상태 관리의 필요성

Buffer Cache는 제한된 메모리 공간에서 수많은 데이터 블록을 관리해야 한다. 새로운 블록을 캐시에 적재하려면 기존 버퍼를 재사용해야 하는데, 이때 "어떤 버퍼를 재사용할 수 있는가"를 결정하는 것이 핵심 문제다.

Oracle은 이 문제를 해결하기 위해 각 버퍼에 **상태(State)**를 부여한다. 버퍼의 상태에 따라 즉시 재사용 가능한지, 디스크에 먼저 기록해야 하는지, 아니면 현재 사용 중이라 건드릴 수 없는지가 결정된다.

## 2. 세 가지 상호 배타적 버퍼 상태

Oracle 공식 문서는 버퍼가 가질 수 있는 세 가지 **상호 배타적(mutually exclusive)** 상태를 정의한다. 상호 배타적이라는 것은 하나의 버퍼가 동시에 두 가지 이상의 상태를 가질 수 없다는 의미다.

### 2.1 Unused (미사용)

정의: 한 번도 사용된 적이 없거나, 현재 사용되지 않아 즉시 사용 가능한 버퍼

특성:

- 데이터베이스가 **가장 쉽게 사용할 수 있는** 버퍼 유형이다.
- 인스턴스가 시작된 직후에는 대부분의 버퍼가 Unused 상태다.
- 새로운 데이터 블록을 적재할 때 Unused 버퍼를 우선적으로 선택한다.
- 어떤 데이터도 포함하지 않으므로 디스크 기록 없이 즉시 덮어쓸 수 있다.

아키텍처적 의의:

Unused 버퍼는 Buffer Cache의 "여유 공간"을 나타낸다. 시스템 운영 초기에는 Unused 버퍼가 많지만, 시간이 지나면서 데이터베이스 활동이 누적되면 대부분의 버퍼가 Clean 또는 Dirty 상태로 전환된다. **Unused 버퍼가 고갈되면 Oracle은 Clean 버퍼를 재사용하거나, Dirty 버퍼를 디스크에 기록한 후 재사용해야 한다.**

### 2.2 Clean (깨끗함)

정의: 이전에 사용되었고, 특정 시점 기준의 읽기 일관성 버전 데이터를 포함하지만, 디스크의 내용과 동일하여 체크포인트 없이 재사용 가능한 버퍼

특성:

- 데이터를 포함하고 있지만 **"깨끗한"** 상태다.

- 메모리의 내용이 디스크의 내용과 **동일**하다.
- 체크포인트(디스크 기록) 없이 데이터베이스가 핀(pin)하고 재사용할 수 있다.
- 새로운 데이터로 덮어써도 데이터 손실이 발생하지 않는다.

#### "깨끗함"의 의미:

Clean이라는 표현은 "변경되지 않았다" 또는 "동기화되어 있다"는 의미다. 버퍼에 데이터가 있지만, 그 데이터는 이미 디스크에 기록된 상태와 동일하므로 별도의 보존 조치 없이 재사용할 수 있다.

#### Clean 버퍼가 되는 경우:

1. 디스크에서 읽어온 후 한 번도 수정되지 않은 블록
2. Dirty 버퍼였지만 DBW가 디스크에 기록을 완료한 블록

## 2.3 Dirty (더러움)

정의: 수정된 데이터를 포함하고 있으며, 아직 디스크에 기록되지 않은 버퍼

#### 특성:

- 메모리의 내용이 디스크의 내용과 **다르다**.
- 재사용하기 전에 반드시 **체크포인트(디스크 기록)**가 필요하다.
- DBW(Database Writer) 프로세스가 디스크에 기록할 때까지 이 상태를 유지한다.
- 디스크에 기록이 완료되면 Clean 상태로 전환된다.

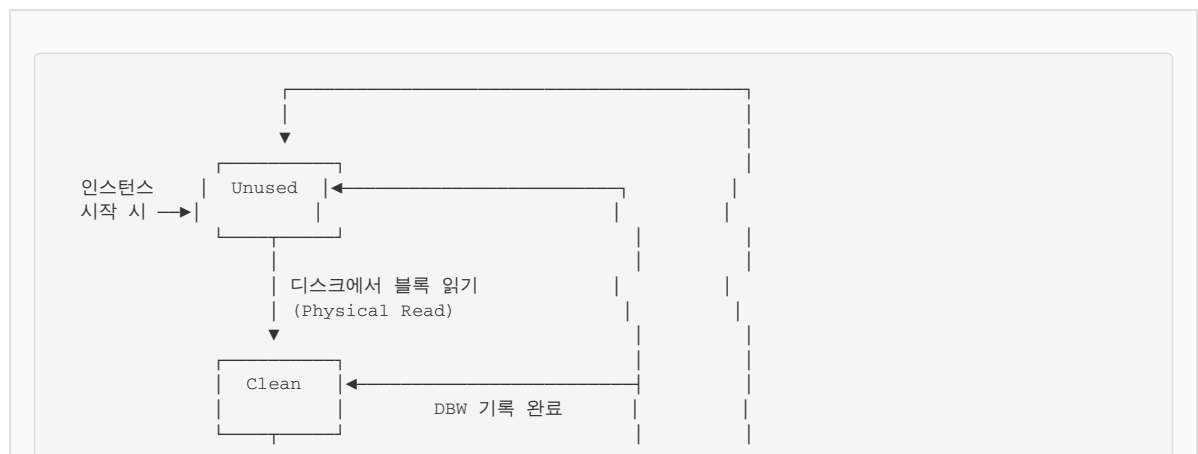
#### Dirty 버퍼의 중요성:

Dirty 버퍼는 **커밋된 트랜잭션의 변경 사항**을 포함할 수 있다. 이 버퍼를 디스크에 기록하지 않고 덮어쓰면 데이터 손실이 발생한다. 따라서 Oracle은 Dirty 버퍼를 재사용하기 전에 반드시 DBW를 통해 디스크에 기록한다.

단, 트랜잭션의 **내구성(Durability)**은 Dirty 버퍼가 아닌 **Redo Log**로 보장된다. 시스템 장애 발생 시 Dirty 버퍼의 내용이 손실되더라도, Redo Log를 통해 복구할 수 있다. 이것이 "Purpose" 섹션에서 설명한 Deferred Write 메커니즘의 핵심이다.

## 3. 버퍼 상태 전이 다이어그램

버퍼는 데이터베이스 활동에 따라 상태가 전이된다.





상태 전이 시나리오:

전이	트리거	설명
Unused → Clean	Physical Read	디스크에서 데이터 블록을 읽어 버퍼에 적재
Clean → Dirty	DML 실행	UPDATE, INSERT, DELETE로 버퍼 내용 수정
Dirty → Clean	DBW 기록	DBW가 Dirty 버퍼를 디스크에 기록 완료
Clean → Unused	버퍼 재사용	LRU에 의해 Cold End의 Clean 버퍼가 새 블록으로 교체될 때 (개념적 전이)

## 4. 버퍼 접근 모드: Pinned vs Free

세 가지 상태와 별개로, 모든 버퍼는 접근 모드(access mode)를 가진다.

### 4.1 Pinned (고정됨)

정의: 사용자 세션이 접근하는 동안 메모리에서 밀려나지 않도록 고정된 버퍼

특성:

- 세션이 버퍼에 접근하는 동안 **LRU 알고리즘에 의해 교체되지 않는다**.
- 여러 세션이 동시에 **Pinned** 버퍼를 수정할 수 없다 (동시성 제어).
- 세션이 버퍼 사용을 완료하면 Free 상태로 전환된다.

**Pinning의 목적:**

Pinning은 버퍼 일관성을 보장한다. 세션 A가 특정 블록을 읽는 도중에 그 버퍼가 다른 데이터로 교체되면 데이터 무결성 문제가 발생한다. Pinning은 세션이 작업을 완료할 때까지 해당 버퍼를 보호한다.

### 4.2 Free (Unpinned, 해제됨)

정의: 어떤 세션도 현재 접근하지 않아 자유롭게 사용 가능한 버퍼

특성:

- LRU 알고리즘에 의해 교체 대상이 될 수 있다.
- 다른 세션이 편하여 사용할 수 있다.

## 5. 버퍼 상태와 재사용 우선순위

새로운 데이터 블록을 Buffer Cache에 적재해야 할 때, Oracle은 다음 우선순위로 버퍼를 선택한다.

우선순위	버퍼 상태	재사용 비용	이유
1순위	Unused + Free	최저	즉시 사용 가능, 어떤 처리도 불필요
2순위	Clean + Free	낮음	데이터가 있지만 디스크와 동일하므로 즉시 덮어쓰기 가능
3순위	Dirty + Free	높음	디스크에 먼저 기록해야 함 (DBW 호출 필요)
사용 불가	Pinned (상태 무관)	-	현재 세션이 사용 중이므로 교체 불가

### 재사용 프로세스:

1. 서버 프로세스가 새 블록을 읽어야 할 때, LRU 리스트의 Cold End부터 탐색 시작
2. Free 상태인 Unused 또는 Clean 버퍼를 찾으면 즉시 사용
3. 탐색 중 Dirty 버퍼를 발견하면 **Dirty List(Write Queue)**로 이동
4. 일정 임계치까지 탐색해도 Free 버퍼를 찾지 못하면 DBW에 신호를 보내 Dirty 버퍼를 디스크에 기록하도록 요청
5. DBW가 Dirty 버퍼를 기록하면 해당 버퍼는 Clean 상태가 되어 재사용 가능

## 6. 백엔드 개발자 관점의 아키텍처적 의의

### 6.1 Dirty 버퍼 누적과 성능 영향

Dirty 버퍼가 과도하게 누적되면 다음과 같은 성능 문제가 발생할 수 있다.

시나리오: 대량 **UPDATE** 트랜잭션

1. 애플리케이션이 대량의 UPDATE를 실행
2. 많은 버퍼가 Dirty 상태로 전환
3. 새로운 쿼리가 실행되어 새 블록을 읽어야 함
4. Clean/Unused 버퍼가 부족하여 DBW에 기록 요청
5. DBW가 Dirty 버퍼를 디스크에 기록하는 동안 대기
6. 쿼리 응답 시간 증가

이 현상은 "**Buffer Busy Waits**" 또는 "**Free Buffer Waits**"라는 대기 이벤트로 나타난다. 백엔드 개발자가 직접 Buffer Cache를 튜닝할 수는 없지만, 대량 **DML** 작업을 배치로 분할하거나 적절한 **COMMIT** 주기를 설정하여 Dirty 버퍼 누적을 완화할 수 있다.

### 6.2 Wherehouse 프로젝트와의 연결

1차 테스트(N+1)에서 25회의 SELECT 쿼리가 실행될 때 버퍼 상태 관점에서 분석하면:

[SELECT 쿼리만 실행하는 경우의 버퍼 상태 변화]

- 1회차: 필요한 블록이 Buffer Cache에 없음  
→ Unused 버퍼 선택  
→ 디스크에서 읽어 적재  
→ 버퍼 상태: Unused → Clean (수정 없으므로 Dirty가 아님)
- 2~25회차: 동일 테이블의 다른 row 조회  
→ 이미 적재된 블록에서 Cache Hit 발생 가능  
→ 해당 버퍼를 Pin하여 읽기  
→ 읽기만 하므로 상태는 Clean 유지

SELECT는 데이터를 수정하지 않으므로 **Dirty 버퍼를 생성하지 않는다**. 따라서 N+1 문제에서 Buffer Cache 관점의 오버헤드는 크지 않다. 문제의 핵심은 버퍼 상태가 아니라 **25번의 쿼리 실행 사이클**에서 발생하는 Connection 경합과 네트워크 왕복이다.

반면, 대량 **UPDATE**가 포함된 배치 작업이라면 Dirty 버퍼 관리가 성능에 영향을 줄 수 있다.

### 6.3 COMMIT 주기와 Dirty 버퍼

Dirty 버퍼는 COMMIT과 직접적인 관계가 없다는 점을 이해해야 한다.

[흔한 오해]

"COMMIT을 하면 Dirty 버퍼가 디스크에 기록된다" → 틀림

[실제 동작]

- COMMIT: Redo Log Buffer → Online Redo Log File (LGWR)
- Dirty 버퍼 기록: Buffer Cache → Data File (DBW, 비동기)

COMMIT은 **Redo Log 기록**만 보장한다. Dirty 버퍼의 디스크 기록은 DBW가 **비동기적으로** 수행하며, COMMIT 시점과 무관하다. 이것이 "Purpose" 섹션에서 설명한 **Lazy Write**의 핵심이다.

## 7. 핵심 정리

상태	디스크 동기화	재사용 가능 여부	재사용 전 필요 작업
Unused	해당 없음	즉시 가능	없음
Clean	동기화됨	즉시 가능	없음
Dirty	동기화 안됨	조건부 가능	DBW가 디스크에 기록

접근 모드	의미	LRU 교체 대상
Pinned	세션이 현재 사용 중	아니오
Free	사용 가능	예

### 설계 원칙:

Buffer States는 "버퍼 재사용의 안전성"을 판단하는 메커니즘이다. Oracle은 이 상태 정보를 기반으로 데이터 손실 없이 효율적으로 제한된 메모리를 관리한다.