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 없는 버퍼로, 인스턴스 시작 직후의 초기 상태

특성:
- 데이터베이스가 **가장 쉽게 사용할 수 있는** 버퍼 유형이다.
- 인스턴스가 시작된 직후에만 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:

Unused는 Buffer Cache의 **초기 상태**를 나타낸다. 인스턴스 시작 직후에는 Unused 버퍼가 대부분이지만, 데이터베이스 활동이 시작되면 빠르게 Clean 또는 Dirty 상태로 전환된다. 일단 사용된 버퍼는 Unused 상태로 되돌아

가지 않으며, 이후에는 Clean과 Dirty 사이에서만 상태가 전이된다. **Unused 버퍼가 고갈되면 Oracle은 Clean 버퍼를 재사용하거나, Dirty 버퍼를 디스크에 기록한 후 재사용해야 한다.**

2.2 Clean (깨끗함)

> 정의: 이전에 사용되었고, 디스크에서 읽어온 후 수정되지 않아
> 메모리 내용이 디스크 내용과 동일한 버퍼

특성:
- 데이터를 포함하고 있지만 **"깨끗한"** 상태다.
- 메모리의 내용이 디스크의 내용과 **동일**하다.
- 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

주의: 읽기 일관성(Read Consistency)과의 구분

Clean 버퍼는 MVCC(Multi-Version Concurrency Control)에서 말하는 CR(Consistent Read) 블록과 다른 개념이다. CR 블록은 Undo 데이터를 적용하여 특정 SCN 시점의 일관된 뷰를 제공하는 반면, Clean 버퍼는 단순히 "디스크와 메모리가 동기화된 상태"를 의미한다.

2.3 Dirty (더러움)

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

특성:
- 메모리의 내용이 디스크의 내용과 **다르다**.
- 재사용하기 전에 반드시 **DBW(Database Writer)에 의한 디스크 기록**이 필요하다.
- DBW가 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.

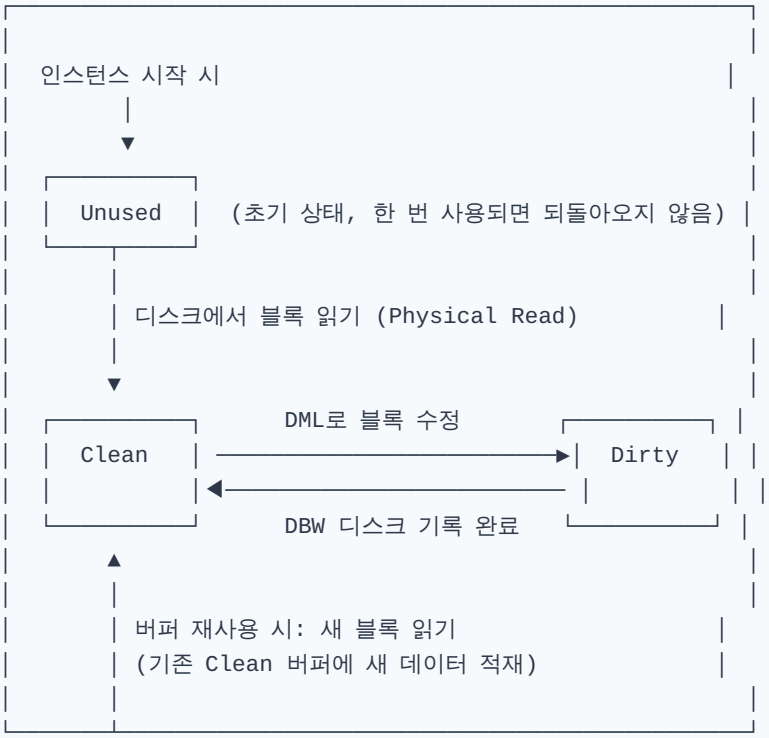
Dirty 버퍼 기록과 Checkpoint의 구분:

개별 Dirty 버퍼가 DBW에 의해 디스크에 기록되는 것은 단순한 **버퍼 기록(buffer write/flush)**이다. 이것을 **Checkpoint**와 혼동해서는 안 된다. Checkpoint는 특정 SCN까지의 모든 Dirty 버퍼를 디스크에 기록하고, Control File 및 Data File 헤더의 SCN을 갱신하는 **시스템 전체 이벤트**다. Checkpoint는 인스턴스 복구 시 Redo 적용 시작점을 결정하는 아키텍처적 경계를 설정하는 반면, 개별 버퍼 기록은 단순히 메모리-디스크 동기화일 뿐이다.

---

3. 버퍼 상태 전이 다이어그램

버퍼는 데이터베이스 활동에 따라 상태가 전이된다. 핵심 원칙은 **Unused는 초기 상태**이며, 일단 사용되면 **Clean과 Dirty 사이에서만 순환**한다는 것이다.



상태 전이 시나리오:

전이	트리거	설명
Unused → Clean	Physical Read	최초로 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 → Clean	버퍼 재사용	LRU에 의해 선택된 Clean 버퍼에 새 블록을 읽어 적재

주의사항:
- Dirty 버퍼는 **반드시 Clean 상태를 거쳐야** 재사용된다. Dirty → Unused 직접 전이는 존재하지 않는다.
- Clean 버퍼가 재사용될 때 "Unused로 되돌아가는" 것이 아니라, 새 블록이 적재되면서 바로 Clean 상태가 된다(읽기의 경우).

---

4. 버퍼 접근 모드: Pinned vs Free

세 가지 상태와 별개로, 모든 버퍼는 **접근 모드(access mode)**를 가진다. 접근 모드는 버퍼 상태(Unused/Clean/Dirty)와 **독립적인 차원**이다.

4.1 Pinned (고정됨)

> 정의: 사용자 세션이 접근하는 동안 메모리에서 밀려나지 않도록 고정된 버퍼

특성:
- 세션이 버퍼에 접근하는 동안 **LRU 알고리즘에 의해 교체되지 않는다**.
- 세션이 버퍼 사용을 완료하면 Free 상태로 전환된다.

Pinning의 목적:

Pinning은 **버퍼 교체 방지**를 담당한다. 세션 A가 특정 블록을 읽는 도중에 그 버퍼가 다른 데이터로 교체되면 데이터 무결성 문제가 발생한다. Pinning은 세션이 작업을 완료할 때까지 해당 버퍼가 LRU에 의해 교체되지 않도록 보호한다.

Pinning과 동시성 제어의 구분:

Pinning 자체는 **동시 수정 방지 메커니즘이 아니다**. 여러 세션이 동시에 같은 버퍼를 수정하지 못하도록 하는 것은 별도의 메커니즘이 담당한다.

- **Buffer Lock (Buffer Pin Mode)**: 버퍼에 대한 접근 모드를 Shared(읽기) 또는 Exclusive(쓰기)로 구분하여 동시 수정 방지

- **Cache Buffers Chains Latch**: 해시 체인 탐색 및 버퍼 헤더 접근 시 직렬화 보장

Pinning은 "이 버퍼를 사용 중이니 교체하지 마라"는 선언이고, 동시성 제어는 Buffer Lock과 Latch가 담당한다. 이 두 개념을 혼동하면 Oracle 동시성 제어 메커니즘 이해에 혼란이 생긴다.

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 핀하여 사용할 수 있다.

---

5. 버퍼 재사용 프로세스

새로운 데이터 블록을 Buffer Cache에 적재해야 할 때, Oracle 서버 프로세스는 다음 과정을 거친다.

5.1 재사용 가능한 버퍼 탐색

서버 프로세스는 LRU 리스트의 Cold End부터 탐색을 시작하여 **Free 상태인 Unused 또는 Clean 버퍼**를 찾는다.

버퍼 상태	Free 여부	즉시 재사용 가능	비고
Unused + Free	○	○	최우선 선택, 어떤 처리도 불필요
Clean + Free	○	○	디스크와 동일하므로 즉시 덮어쓰기 가능
Dirty + Free	○	×	서버 프로세스가 직접 재사용 불가
Pinned (상태 무관)	×	×	현재 세션이 사용 중이므로 탐색 대상 제외

5.2 Dirty 버퍼만 남은 경우의 처리

서버 프로세스는 **Dirty 버퍼를 직접 재사용하지 않는다**. Clean/Unused 버퍼를 찾지 못하면 다음 과정을 거친다.

> [Dirty 버퍼 처리 프로세스]
>
> 1. 서버 프로세스가 LRU 리스트 탐색 중 Clean/Unused 버퍼를 찾지 못함
> 2. 탐색 중 발견한 Dirty 버퍼들을 Dirty List(Write Queue)로 이동
> 3. 일정 임계치까지 탐색해도 Free 버퍼를 찾지 못하면 DBW에 신호 전송
> 4. DBW가 Dirty List의 버퍼들을 디스크에 기록
> 5. 기록 완료된 버퍼는 Dirty → Clean으로 상태 전환
> 6. 서버 프로세스가 Clean으로 전환된 버퍼를 재사용

이 과정에서 서버 프로세스는 DBW의 기록 완료를 **대기(wait)**해야 한다. 이 대기 시간이 **"Free Buffer Waits"** 대기 이벤트로 나타난다.

5.3 재사용 비용 관점

시나리오	재사용 비용	발생하는 작업
Unused/Clean 버퍼 존재	최저	즉시 버퍼 할당
Dirty 버퍼만 존재	높음	DBW 신호 → 디스크 I/O → Clean 전환 → 버퍼 할당

---

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 분할**하거나 **적절한 COMMIT 주기를 설정**하여 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 UPDATE가 포함된 배치 작업**이라면 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 후 Clean 전환

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.

용어 구분:
- **버퍼 기록(Buffer Write/Flush)**: 개별 Dirty 버퍼를 DBW가 디스크에 기록하는 작업
- **Checkpoint**: 특정 SCN까지의 모든 Dirty 버퍼를 기록하고, Control File/Data File 헤더를 갱신하는 시스템 이벤트

본 문서는 Oracle 19c Database Concepts 공식 문서를 기반으로 백엔드 개발자 관점에서 재구성한 학습 자료입니다.