Oracle Buffer Reads 학습 문서

문서 개요

- **작성 목적**: Oracle Memory Architecture 중 Buffer Reads 메커니즘 학습
- **학습 기준 문서**: Oracle Database Concepts 19c - Memory Architecture 챕터
- **공식 문서 URL**: https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/memory-architecture.html#GUID-1A40F9B9-EB2F-4060-9007-7B26C033A774
- **프로젝트 연관성**: Wherehouse 프로젝트 N+1 최적화 성능 테스트 결과 해석

---

1. Buffer Reads: 버퍼 캐시에서 데이터를 읽는 메커니즘

1.1 왜 Buffer Reads를 이해해야 하는가 (Why)

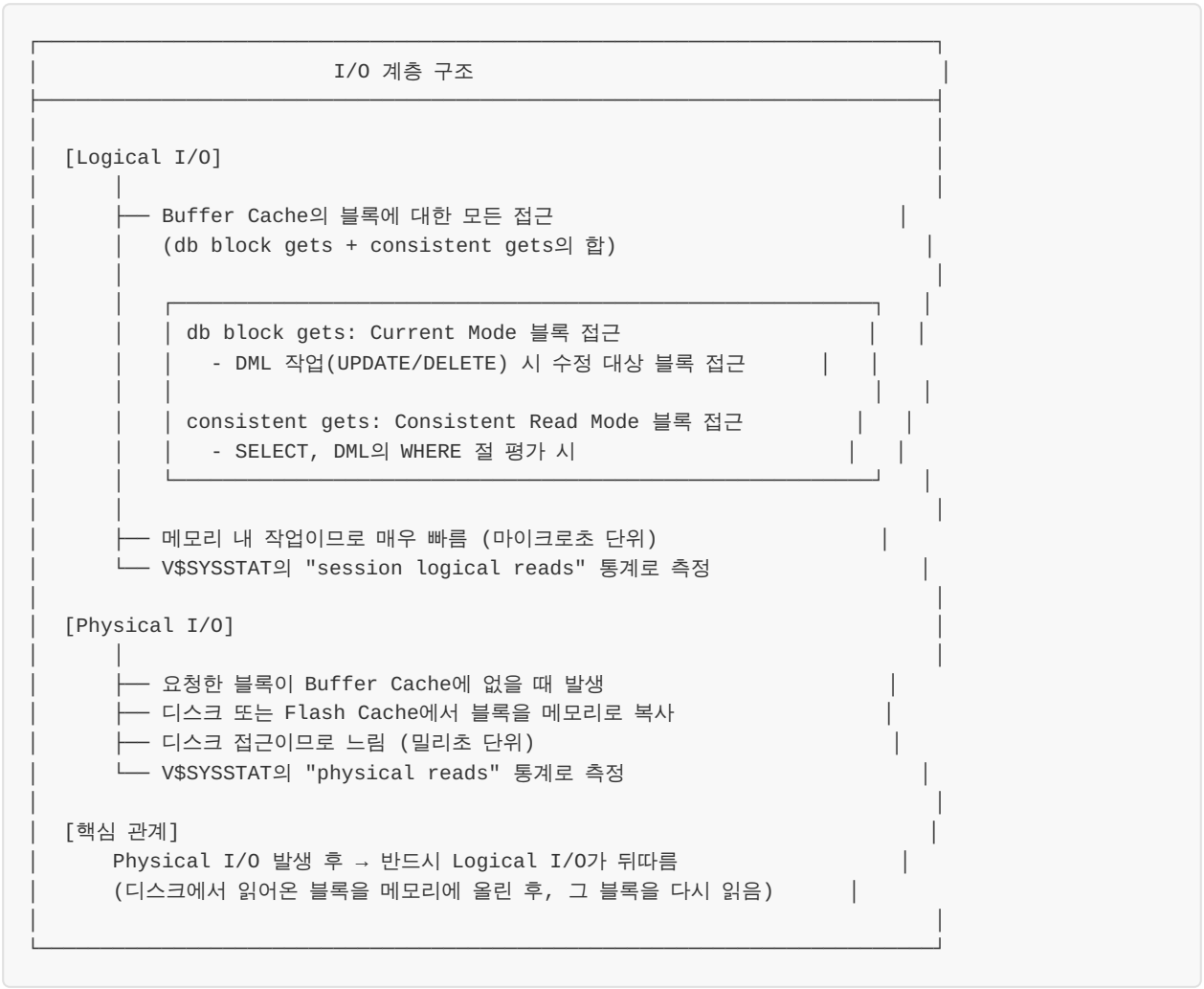
Buffer Reads는 Oracle이 클라이언트 요청에 응답하기 위해 데이터 블록을 어디서, 어떻게 가져오는지를 결정하는 핵심 메커니즘이다. 이 메커니즘을 이해해야 하는 이유는 Wherehouse 프로젝트의 성능 테스트 결과를 아키텍처 레벨에서 해석할 수 있기 때문이다.

테스트 데이터에서 "RDB 조회 시간"이라고 측정한 항목의 실체가 바로 이 Buffer Reads 과정에서 발생하는 Logical I/O와 Physical I/O의 합산이다. 1차 테스트에서 RDB 시간 비율이 6.2%에 불과했다는 것은, 실제 데이터 접근 시간보다 **그 외의 오버헤드**(Connection 획득 대기, 네트워크 왕복 등)가 훨씬 컸다는 의미다. 이 현상을 정확히 설명하려면 Buffer Reads의 내부 동작을 알아야 한다.

1.2 Buffer Reads의 동작 원리 (How)

1.2.1 핵심 개념: Logical I/O vs Physical I/O

Oracle에서 I/O는 두 가지 계층으로 구분된다:



이 구분이 중요한 이유는 **Buffer Cache Hit Ratio**라는 핵심 성능 지표의 근거가 되기 때문이다:

```
Buffer Cache Hit Ratio = 1 - (Physical Reads / Logical Reads)
```

이 비율이 높을수록 디스크 접근 없이 메모리에서 데이터를 가져오는 비율이 높다는 의미다.

1.2.2 Buffer 검색 순서 (Search Order)

클라이언트가 데이터를 요청하면, Server Process는 다음 순서로 버퍼를 검색한다:


Buffer 검색 순서 (3단계)
[Client Request: SELECT * FROM REVIEW_STATISTICS WHERE PROPERTY_ID = 'P001']
[1단계] Buffer Cache에서 전체 버퍼 검색
Server Process가 요청된 블록의 DBA(Data Block Address)를
Hash Function에 입력하여 Hash Bucket을 찾음
Hash Bucket 내의 Buffer Header Chain을 순회하며
해당 블록이 메모리에 존재하는지 확인
[Cache Hit] → Logical Read 수행 → 완료
[Cache Miss] → 2단계로 진행
▼ (Cache Miss인 경우)
[2단계] Flash Cache LRU 리스트에서 Buffer Header 검색
⚠️ 선택적 단계: Flash Cache가 활성화된 경우에만 해당
(Oracle Linux, Solaris의 Exadata/특정 환경에서만 사용 가능)
AWS RDS, Azure, 일반 온프레미스 환경에서는 1단계→3단계 직행
Flash Cache는 SSD 기반의 2차 캐시 계층
Buffer Body만 Flash Cache에 저장되고,
Buffer Header는 메인 메모리의 LRU 리스트에 유지됨
[Header Found] → Flash Cache에서 Body 읽기
(Optimized Physical Read)
→ In-Memory Cache로 복사
→ Logical Read 수행 → 완료
[Header Not Found] → 3단계로 진행
▼ (Flash Cache에도 없는 경우)
[3단계] Data File에서 Physical Read
Server Process가 OS에 I/O 요청
디스크에서 블록을 읽어 Buffer Cache로 복사
Logical Read 수행 → 완료
이 과정에서 발생하는 Wait Event:
- db file sequential read (단일 블록, 인덱스 스캔 시)
- db file scattered read (다중 블록, Full Table Scan 시)

1.2.3 Cache Hit vs Cache Miss의 성능 차이

공식 문서에서 **"accessing data through a cache hit is faster than through a cache miss"**라고 명시한 부분의 정량적 의미:

```
┌──────────────────────────────────────────────────────────────────┐
│                    접근 방식별 지연 시간 비교                    │
├──────────────────────┬──────────────────────┬────────────────────┤
│      접근 방식       │     지연 시간        │        비고        │
├──────────────────────┼──────────────────────┼────────────────────┤
│  Cache Hit           │   0.01 ~ 0.1 ms      │  메모리 접근만 발생 │
│  (Logical I/O only)  │                      │  CPU bound 작업    │
├──────────────────────┼──────────────────────┼────────────────────┤
│  Flash Cache Hit     │   0.1 ~ 1 ms         │  SSD 접근          │
│  (Optimized Physical)│                      │  HDD 대비 10~100배 빠름 │
├──────────────────────┼──────────────────────┼────────────────────┤
│  Cache Miss          │   5 ~ 20 ms          │  HDD 접근          │
│  (Full Physical I/O) │                      │  I/O bound 작업    │
└──────────────────────┴──────────────────────┴────────────────────┘
```

이 차이가 **N+1 문제에서 증폭되는 메커니즘**:

```
[N+1 쿼리 25회 실행 시 Buffer Reads 시나리오]

시나리오 A: 모든 블록이 Cache Hit인 경우
    25회 × 0.05ms (Logical I/O) = 1.25ms

시나리오 B: 모든 블록이 Cache Miss인 경우 (Cold Start)
    25회 × 10ms (Physical I/O) = 250ms

시나리오 C: 현실적 혼합 (80% Hit, 20% Miss)
    20회 × 0.05ms + 5회 × 10ms = 51ms

→ 동일한 25회 쿼리라도 Buffer Cache 상태에 따라
   1.25ms ~ 250ms까지 200배의 성능 차이 발생 가능
```

1.2.4 Temp File 읽기의 특수성

공식 문서에서 중요하게 언급한 예외 케이스:

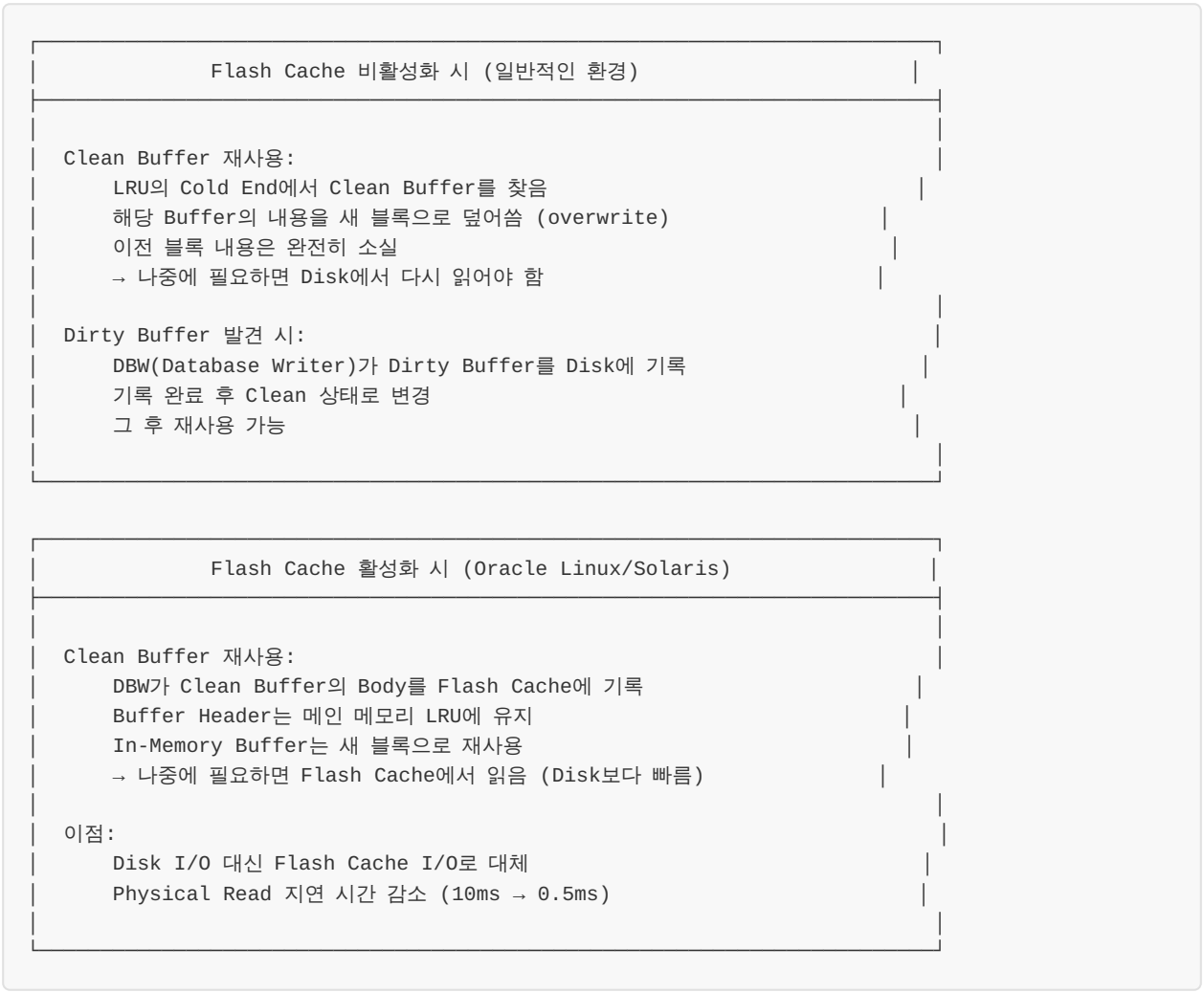
> "Reads from a temp file occur when insufficient memory forces the database to write data to a temporary table and read it back later. These physical reads bypass the buffer cache and do not incur a logical I/O."

이것이 의미하는 바:

프로젝트 연관성: Wherehouse의 주거지 추천 쿼리에서 ORDER BY나 복잡한 집계가 포함된 경우, PGA의 Sort Area가 부족하면 Temp File I/O가 발생할 수 있다. 이 경우 V$SYSSTAT의 "physical reads direct temporary" 통계가 증가한다.

1.2.5 Buffer 확보를 위한 Aging Out 메커니즘

Buffer Cache에 새 블록을 읽어와야 하는데 **Unused Buffer가 부족한 경우**, Oracle은 기존 버퍼를 Aging Out해야 한다. 공식 문서에서 설명한 두 가지 시나리오:


Flash Cache 비활성화 시 (일반적인 환경)
Clean Buffer 재사용:
LRU의 Cold End에서 Clean Buffer를 찾음
해당 Buffer의 내용을 새 블록으로 덮어씀 (overwrite)
이전 블록 내용은 완전히 소실
→ 나중에 필요하면 Disk에서 다시 읽어야 함
Dirty Buffer 발견 시:
DBW(Database Writer)가 Dirty Buffer를 Disk에 기록
기록 완료 후 Clean 상태로 변경
그 후 재사용 가능
Flash Cache 활성화 시 (Oracle Linux/Solaris)
Clean Buffer 재사용:
DBW가 Clean Buffer의 Body를 Flash Cache에 기록
Buffer Header는 메인 메모리 LRU에 유지
In-Memory Buffer는 새 블록으로 재사용
→ 나중에 필요하면 Flash Cache에서 읽음 (Disk보다 빠름)
이점:
Disk I/O 대신 Flash Cache I/O로 대체
Physical Read 지연 시간 감소 (10ms → 0.5ms)

1.3 프로젝트 테스트 데이터와의 연결 (So What)

1.3.1 1차 테스트(N+1) 결과 재해석

```
┌──────────────────────────────────────────────────────────────────────┐
│  1차 테스트 결과                                                     │
│  ──────────────────────────────────────────────────────────────────  │
│  총 소요 시간: 5,531ms                                               │
│  RDB 조회 시간: 343ms                                                │
│  RDB 시간 비율: 6.2%                                                 │
│  쿼리 실행 횟수: 25회                                                │
│  평균 쿼리당 시간: 343ms ÷ 25 = 13.7ms                               │
└──────────────────────────────────────────────────────────────────────┘

[Buffer Reads 관점에서의 해석]

쿼리당 13.7ms가 의미하는 것:
    - 순수 Physical I/O만 발생했다면: 10~20ms 범위 (일치)
    - Cache Hit가 많았다면: 1ms 미만이어야 함

→ 추론: REVIEW_STATISTICS 테이블의 블록들이 Buffer Cache에
        충분히 캐싱되지 않은 상태였을 가능성 높음
         (Cold Start 또는 Buffer Cache 크기 부족)

[나머지 94%는 어디서 소요되었는가?]

5,531ms - 343ms = 5,188ms가 DB 외부에서 소요

이 시간의 구성 요소:
    1. 네트워크 왕복 시간 × 25회
    2. JDBC 드라이버 처리 시간 × 25회
    3. Hibernate 영속성 컨텍스트 처리 × 25회
    4. HikariCP Connection 획득 대기 시간 (Waiting 최대 9건)
    5. Server Process의 Parse 시간 × 25회

→ Buffer Reads 자체보다 "Buffer Reads를 하기 위한 준비 과정"이
   병목의 주요 원인
```

1.3.2 Buffer Cache Hit Ratio 추정

테스트 데이터로부터 역산:

```
[가정]
- 단일 쿼리가 접근하는 평균 블록 수: 100개
- Logical I/O 1회 시간: 0.05ms
- Physical I/O 1회 시간: 10ms

[1차 테스트 - 쿼리당 13.7ms]
만약 Hit Ratio = X라면:
    총 시간 = (Hit 블록 수 × Logical I/O 시간) + (Miss 블록 수 × Physical I/O 시간)

     100 × X × 0.05ms + 100 × (1-X) × 10ms = 13.7ms
     5X + 1000 - 1000X = 13.7
     1000 - 995X = 13.7
     -995X = -986.3
     X ≈ 0.991 (약 99%)

→ 계산상 Hit Ratio가 99%인데도 13.7ms가 소요된다면,
   나머지 1%의 Physical I/O(1개 블록)가 전체 시간을 지배함

[시사점]
Buffer Cache Hit Ratio가 99%로 높아도,
1%의 Physical I/O가 전체 응답 시간의 대부분을 차지할 수 있음.
이것이 "Buffer Cache Hit Ratio는 높은데 왜 느리지?"라는
흔한 의문에 대한 아키텍처적 답변임.
```

1.3.3 면접 답변 구성

질문: "N+1 쿼리가 왜 성능에 영향을 주나요?"

```
[합격선 답변 구조]

1. Buffer Reads 관점에서의 영향:
   "N+1 쿼리는 동일한 데이터에 대해 25번의 개별 Buffer Read 요청을
    발생시킵니다. 첫 번째 쿼리에서 Physical I/O가 발생하면 블록이
     Buffer Cache에 올라가므로 이후 쿼리는 Cache Hit가 될 수 있습니다.

    그러나 문제는 Buffer Read 자체보다 그것을 수행하기 위한
    오버헤드에 있습니다."

2. 오버헤드 구체화:
   "25번의 쿼리는 각각:
     - Parse 단계를 거쳐야 하고 (Library Cache 조회)
     - Buffer Cache 접근을 위해 cache buffers chains latch를 획득해야 하며
     - 결과를 클라이언트로 전송하는 네트워크 왕복이 필요합니다.

    제 테스트에서 RDB 시간 비율이 6.2%에 불과했던 것은,
    실제 Buffer Read 시간보다 이러한 부수적 오버헤드가
     15배 이상 컸다는 의미입니다."

3. 본인 데이터 연결:
   "HikariCP Waiting이 최대 9건까지 발생한 것은,
     25번의 Buffer Read 사이클 동안 Connection이 계속 점유되어
    다른 요청들이 Connection을 획득하지 못했기 때문입니다."
```

1.4 추가 학습 포인트

Buffer Reads를 완전히 이해하려면 다음 연관 개념도 학습이 필요하다:

1. **Buffer States (Unused/Clean/Dirty)** - 버퍼가 재사용 가능한 조건
2. **LRU Algorithm** - 어떤 버퍼가 Aging Out 대상이 되는지
3. **Touch Count** - 버퍼의 "Hot/Cold" 판정 메커니즘
4. **db file sequential read vs db file scattered read** - Wait Event와의 연결

---

2. FLUSH BUFFER_CACHE: Buffer Cache 강제 비우기

2.1 정의와 목적 (Why)

`FLUSH BUFFER_CACHE`는 Oracle Buffer Cache에 있는 모든 버퍼를 강제로 비우는 명령어다. 정확한 구문은 다음과 같다:

```
ALTER SYSTEM FLUSH BUFFER_CACHE;
```

이 명령어가 존재하는 이유는 **성능 테스트의 재현성** 때문이다. Buffer Cache 상태에 따라 동일한 쿼리라도 응답 시간이 200배까지 차이날 수 있다고 앞서 설명했다. 테스트 결과를 비교하려면 매번 동일한 "Cold Start" 상태에서 시작해야 하는데, 이를 인위적으로 만들어주는 것이 FLUSH BUFFER_CACHE다.

2.2 동작 원리 (How)

2.2.1 내부 동작 메커니즘

```
┌──────────────────────────────────────────────────────────────────────┐
│            ALTER SYSTEM FLUSH BUFFER_CACHE 실행 시 동작              │
├──────────────────────────────────────────────────────────────────────┤
│                                                                      │
│  [핵심 특성] Dirty Buffer는 flush 대상이 아님                        │
│      이 명령어는 Clean Buffer만 무효화함                             │
│      Dirty Buffer는 Buffer Cache에 그대로 유지됨                     │
│                                                                      │
│  [Clean Buffer 처리]                                                 │
│      Clean Buffer를 "Free" 상태로 전환                               │
│      Buffer Header의 블록 매핑 정보 제거                             │
│      LRU 리스트에서 Free List로 이동                                 │
│                                                                      │
│  [Dirty Buffer 처리]                                                 │
│      변경 없음 - Buffer Cache에 그대로 남아있음                      │
│      DBWn의 일반적인 Checkpoint 메커니즘에 의해서만 기록됨           │
│                                                                      │
│  [결과 상태]                                                         │
│      Clean Buffer만 무효화된 "Partial Cold" 상태                     │
│      Dirty Buffer가 있던 블록은 여전히 Cache Hit 가능                │
│                                                                      │
└──────────────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────────────┐
│            완전한 Cold Start 시뮬레이션을 위한 올바른 절차           │
├──────────────────────────────────────────────────────────────────────┤
│                                                                      │
│  -- 1단계: Dirty Buffer를 디스크에 기록하여 Clean 상태로 전환        │
│  ALTER SYSTEM CHECKPOINT;                                            │
│                                                                      │
│  -- 2단계: Clean Buffer 무효화                                       │
│  ALTER SYSTEM FLUSH BUFFER_CACHE;                                    │
│                                                                      │
│  이 순서로 실행해야 Buffer Cache가 완전히 비워진 상태가 됨           │
│                                                                      │
└──────────────────────────────────────────────────────────────────────┘
```

2.2.2 Shared Pool Flush와의 차이

Buffer Cache만 비우는 것과 Shared Pool까지 비우는 것은 다른 의미를 갖는다:

```
┌──────────────────────────────────────────────────────────────────────┐
│                        Flush 명령어 비교                             │
├──────────────────────────────┬───────────────────────────────────────┤
│ 명령어                       │ 영향 범위                             │
├──────────────────────────────┼───────────────────────────────────────┤
│ ALTER SYSTEM                 │ Buffer Cache만 비움                   │
│ FLUSH BUFFER_CACHE;          │ → 다음 쿼리에서 Physical I/O 발생     │
│                              │ → 실행 계획(Library Cache)은 유지     │
│                              │ → Parse는 Soft Parse 가능             │
├──────────────────────────────┼───────────────────────────────────────┤
│ ALTER SYSTEM                 │ Shared Pool 비움                      │
│ FLUSH SHARED_POOL;           │ → Library Cache의 실행 계획 제거      │
│                              │ → 다음 쿼리에서 Hard Parse 발생       │
│                              │ → Data Dictionary Cache도 비워짐      │
├──────────────────────────────┼───────────────────────────────────────┤
│ 두 명령어 모두 실행          │ 완전한 Cold Start 시뮬레이션          │
│                              │ → Physical I/O + Hard Parse 모두 발생 │
│                              │ → 인스턴스 재시작과 유사한 상태       │
└──────────────────────────────┴───────────────────────────────────────┘
```

2.3 프로젝트 테스트에서의 활용 (So What)

2.3.1 테스트 재현성 확보

Wherehouse 프로젝트의 Before/After 비교에서 이 명령어가 필요한 이유:

```
[문제 시나리오: FLUSH 없이 테스트]

테스트 1회차 (Origin Code):
    → 첫 실행이므로 Buffer Cache가 비어있음 (Cold)
    → Physical I/O 다수 발생
    → 응답 시간: 5,531ms

테스트 2회차 (Bulk Code):
    → 1회차에서 읽은 블록들이 Buffer Cache에 남아있음 (Warm)
    → Cache Hit 발생
    → 응답 시간: 3,200ms (실제보다 빠르게 측정됨)

테스트 3회차 (Chunk Code):
    → 2회차까지의 블록들이 더 많이 캐싱됨 (Hot)
    → 더 높은 Cache Hit
    → 응답 시간: 2,100ms (실제보다 더 빠르게 측정됨)

→ 결론: Buffer Cache 상태가 다르므로 공정한 비교 불가
```

```
[해결 시나리오: 매 테스트 전 CHECKPOINT + FLUSH 실행]

테스트 1회차 (Origin Code):
    ALTER SYSTEM CHECKPOINT;
    ALTER SYSTEM FLUSH BUFFER_CACHE;
    → Cold Start 상태
    → 응답 시간: 5,531ms

테스트 2회차 (Bulk Code):
    ALTER SYSTEM CHECKPOINT;
    ALTER SYSTEM FLUSH BUFFER_CACHE;
    → Cold Start 상태 (동일 조건)
    → 응답 시간: 6,001ms

테스트 3회차 (Chunk Code):
    ALTER SYSTEM CHECKPOINT;
    ALTER SYSTEM FLUSH BUFFER_CACHE;
    → Cold Start 상태 (동일 조건)
    → 응답 시간: 4,434ms

→ 결론: 동일한 Cache 상태에서 측정했으므로 공정한 비교 가능
```

2.3.2 테스트 시 권장 절차

```
-- 1. Dirty Buffer를 디스크에 기록 (Clean 상태로 전환)
ALTER SYSTEM CHECKPOINT;

-- 2. Buffer Cache 비우기 (Physical I/O 조건 동일화)
ALTER SYSTEM FLUSH BUFFER_CACHE;

-- 3. Shared Pool 비우기 (Parse 조건 동일화) - 선택사항
ALTER SYSTEM FLUSH SHARED_POOL;

-- 4. 시스템 통계 초기화 (측정 기준점 설정)
-- V$SYSSTAT 값을 기록해두거나, AWR 스냅샷 생성

-- 5. 테스트 실행
-- JMeter 또는 애플리케이션에서 부하 인가

-- 6. 결과 수집
-- V$SYSSTAT 변화량, V$SQL 통계 조회
```

2.3.3 주의사항: 운영 환경에서의 금기

```
┌─────────────────────────────────────────────────────────────┐
│ ⚠️ 경고: 운영 환경에서 FLUSH BUFFER_CACHE 실행 금지          │
├─────────────────────────────────────────────────────────────┤
│                                                             │
│ [발생하는 문제]                                             │
│                                                             │
│ 1. 즉각적인 성능 저하                                       │
│    - 모든 쿼리가 Physical I/O 발생                          │
│    - 응답 시간 10~100배 증가                                │
│    - 사용자 체감 지연 급증                                  │
│                                                             │
│ 2. I/O 폭주                                                 │
│    - 디스크 I/O 대역폭 포화                                 │
│    - 스토리지 병목 발생                                     │
│    - 다른 인스턴스/서비스에도 영향                          │
│                                                             │
│ 3. 연쇄적 장애 가능성                                       │
│    - Connection Timeout 급증                                │
│    - 애플리케이션 에러 발생                                 │
│    - 서비스 장애로 확대                                     │
│                                                             │
│ [허용되는 상황]                                             │
│    - 개발/테스트 환경에서 성능 테스트 시                    │
│    - DBA가 특수한 목적으로 통제된 환경에서 실행             │
│                                                             │
└─────────────────────────────────────────────────────────────┘
```

2.4 면접에서의 활용

질문: "성능 테스트의 재현성을 어떻게 확보했나요?"

```
[합격선 답변]

"동일한 조건에서 비교하기 위해 각 테스트 전에
ALTER SYSTEM CHECKPOINT와 ALTER SYSTEM FLUSH BUFFER_CACHE를
순차적으로 실행하여 Buffer Cache를 완전히 초기화했습니다.

CHECKPOINT는 Dirty Buffer를 디스크에 기록하여 Clean 상태로 만들고,
FLUSH BUFFER_CACHE는 Clean Buffer를 무효화합니다.
FLUSH만 실행하면 Dirty Buffer가 남아 완전한 Cold Start가 아닙니다.

이렇게 하지 않으면 이전 테스트에서 캐싱된 블록들이
다음 테스트에 영향을 주어, Cache Hit Ratio가 달라지고
공정한 비교가 불가능합니다.

다만 이 명령어들은 운영 환경에서는 절대 실행하면 안 됩니다.
모든 쿼리가 Physical I/O를 발생시켜
즉각적인 성능 저하를 유발하기 때문입니다."
```

2.5 연관 개념

FLUSH BUFFER_CACHE를 이해했다면, 다음 개념들도 연결된다:

개념	연관성
Checkpoint	Dirty Buffer를 디스크에 기록하는 메커니즘. FLUSH 전에 실행해야 완전한 Cold Start 가능
Instance Recovery	인스턴스 재시작 시 Buffer Cache가 비어있는 상태로 시작하는 것과 동일한 효과
Warm-up 시간	FLUSH 후 Cache가 다시 "Hot" 상태가 되기까지 걸리는 시간

테스트 문서에 이 명령어 실행 여부를 명시하면 피드백에서 요구한 **"테스트 환경과 조건 명시"** 항목을 충족할 수 있다.

---

3. 참고 자료

3.1 Oracle 공식 문서

- **Memory Architecture**: https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/memory-architecture.html
- **Database Buffer Cache**: https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/memory-architecture.html#GUID-4FF66585-E469-4631-9225-29D75594CD14
- **Buffer I/O**: https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/memory-architecture.html#GUID-D1429BAA-6543-4B34-93DB-C8F33D497B53

3.2 관련 V$ 동적 성능 뷰

뷰 이름	용도
V$SYSSTAT	session logical reads, physical reads 통계
V$BUFFER_POOL_STATISTICS	Buffer Pool별 Hit Ratio 계산
V$BH	개별 Buffer Header 상태 조회

3.3 프로젝트 연관 문서

- Oracle_아키텍처_학습_로드맵_v2.docx

- Oracle_아키텍처_집중학습_섹션_선정_가이드.docx
- 피드백_반영_학습_가이드.docx
- DBMS_섹션_누락사항_피드백.txt

---

4. 문서 이력

버전	일자	변경 내용
1.0	2024-12	초안 작성 - Buffer Reads, FLUSH BUFFER_CACHE 섹션
1.1	2024-12	기술 검토 반영 - FLUSH 동작 메커니즘 수정(Dirty Buffer 미처리 명시), Logical I/O 정의 명확화, 계산 전개 보완, Latch 표현 정확화, Flash Cache 적용 환경 명시, 테스트 절차에 CHECKPOINT 추가