**6장 DML 튜닝**

**2. Direct Path I/O 활용**

**2.1** Direct Path I/O

- 일반적인 블록 I/O는 읽고자 하는 블록을 버퍼캐시에 먼저 가서 찾아보고, 찾지 못하면 디스크에서

읽는다. 데이터를 변경할 때도 버퍼캐시를 먼저 간다. 버퍼블록에 변경을 하고 나면, DBWR 프로세

스가 변경된 블록(Dirty 블록)들을 주기적으로 찾아 데이터파일에 반영한다.

- 버퍼캐시를 이용해 자주 읽는 블록에 대한 반복적인 I/O를 줄임으로써 전반적인 성능을 높인다.

- 하지만, 대량 데이터를 읽고 쓸 때 건건이 버퍼캐시를 탐색한다면 성능이 낮아질뿐더러 대량 블록을

건건이 디스크로부터 버퍼캐시에 적재하고나서 읽어야 하므로 부담이 크다. ( 적재한 블록을 재사용

할 가능성? -> Full Scan 위주의 대용량 처리 프로그램은 보통 재사용성이 낮다. )

- 오라클은 버퍼캐시를 경유하지 않고 바로 데이터 블록을 읽고 쓸 수 있는 **Direct Path I/O** 기능을

제공함

- Direct Path I/O 기능이 작동하는 경우

① 병렬 쿼리로 Full Scan을 수행할 때

- 쿼리문에 parallel 또는 parallel\_index 힌트를 사용하면 지정한 병렬도만큼 병렬 프로세스가

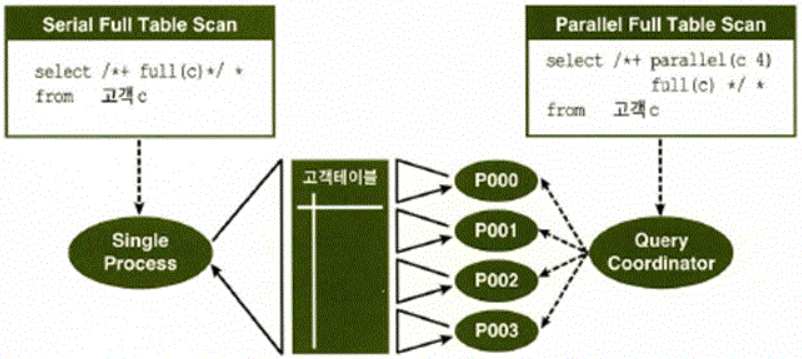
떠서 동시에 작업을 진행 ( 작은 단위로 나누어 동시에 처리함. )

**select /\*+ full(t) parallel(t 4) \*/ \***

**from big\_table t;**

**select /\*+ index\_ffs(t big\_table\_x1) parallel\_index(t big\_table\_x1 4) \*/ count(\*)**

**from big\_table t;**



- 대량 테이블을 병렬도 4로 full scan을 하면 4개의 parallel 세션이 서로 다른 부분의 데이터를

읽어 처리하고 최종적으로 쿼리 전체를 총괄하는 조정 세션에 결과를 전달한다.

- 버퍼캐시를 탐색하지 않고 디스크로부터 버퍼캐시에 적재하지 않으므로 빨라짐.

② 병렬 DML을 수행할 때

③ Direct Path Insert를 수행할 때

④ Temp 세그먼트 블록들을 읽고 쓸 때

⑤ direct 옵션을 지정하고 export를 수행할 때

⑥ nocache 옵션을 지정한 LOB 컬럼을 읽을 때

**2.2** Direct Path Insert

- 일반적인 INSERT 방식

① 데이터를 입력할 수 있는 블록을 Freelist(테이블 HWM 아래쪽에 있는 블록 중 데이터 입력이

가능한 블록의 목록)에서 찾는다.

② Freelist에서 할당받은 블록을 버퍼캐시에서 찾는다.

③ 버퍼캐시에 없으면, 데이터파일에서 읽어 버퍼캐시에 적재한다.

④ INSERT 내용을 Undo 세그먼트에 기록한다.

⑤ INSERT 내용을 Redo 로그에 기록한다.

- Direct Path Insert 방식

① INSERT … SELECT 문에 append 힌트 사용

② parallel 힌트를 이용해 병렬 모드로 INSERT

③ direct 옵션을 지정하고 SQL\*Loader(sqlldr)로 데이터 적재

④ create table … as select문 수행

- Direct Path Insert 방식이 **빠른 이유**

① Freelist를 참조하지 않고 HWM 바깥 영역에 데이터를 순차적으로 입력

② 블록을 버퍼캐시에서 탐색하지 않는다

③ 버퍼캐시에 적재하지 않고, 데이터파일에 직접 기록한다.

④ Undo 로깅을 안 한다.

⑤ nologging 모드로 전환한 상태에서 수행하면 Redo 로깅을 하지 않을 수 있다.

(일반 INSERT문에서 로깅하지 않게 하는 방법 X)

- Direct Path Insert 사용 시 주의할 점

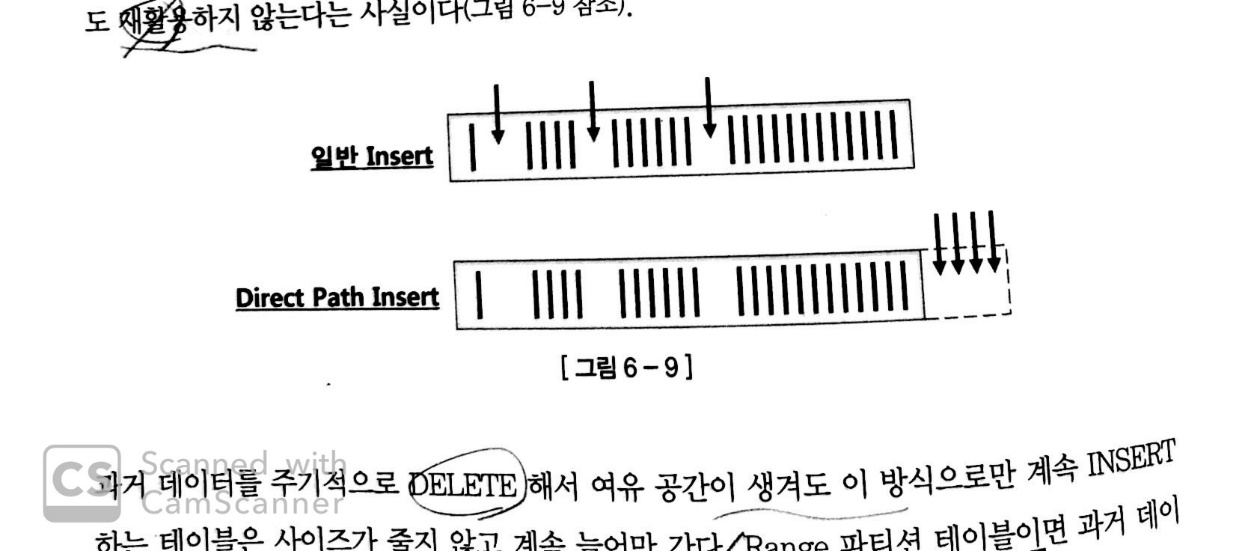
① Exclusive 모드 TM Lock이 걸림. 커밋하기 전까지 다른 트랜잭션은 해당 테이블에 DML 수행 X

-> 트랜잭션이 빈번한 주간에는 절대 사용해서는 안된다!!

② Freelist를 조회하지 않고 HWM 바깥 영역에 입력하므로 테이블에 여유 공간이 있어도 재활용

하지 않는다. 과거 데이터를 주기적으로 DELETE 해서 여유 공간이 생겨도 바깥 영역에 입력되

므로 테이블 사이즈는 계속 늘어만 간다.



**2.3** 병렬 DML

- INSERT는 append 힌트를 이용해 Direct Path 방식으로 유도할 수 있지만 UPDATE, DELETE는 불가능

- 병렬 처리는 대용량 데이터가 전제이므로 오라클은 병렬 DML에 항상 Direct Path Write 방식을

**alter session enable parallel dml;**

사용

- DML 병렬로 처리하려면 병렬 DML 활성화!!

- 활성화 후에 DML문에 힌트를 사용하면 SELECT, UPDATE, DELETE 모두 병렬로 진행

- 실수로 활성화 하지 않고 힌트를 기술한 쿼리를 실행하면 대상 레코드를 찾는 작업은 병렬로

진행하지만 추가, 변경, 삭제는 QC(조정쿼리)가 혼자 담당하므로 병목이 생긴다.

- 병렬 INSERT는 append 힌트를 지정하지 않아도 Direct Path Insert 방식을 사용한다. 하지만 병렬

**insert /\*+ append parallel(c 4) \*/ into 고객 c;**

DML이 작동하지 않을 경우를 대비해 append 힌트를 같이 사용하는게 좋다.

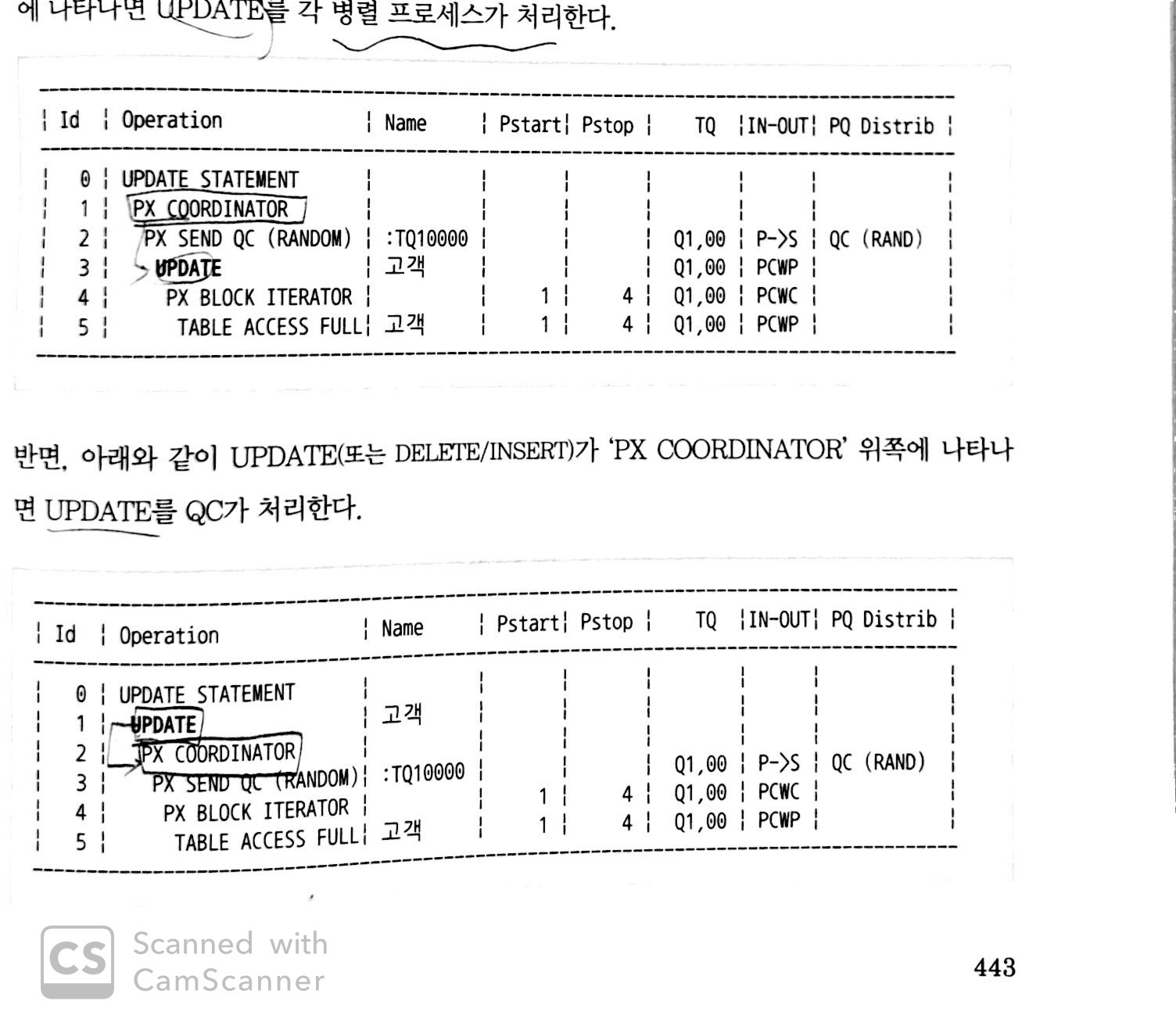
- 12c부터는 enable\_parallel\_dml 힌트도 지원

**insert /\*+ enable\_parallel\_dml parallel(c 4) \*/ into 고객 c;**

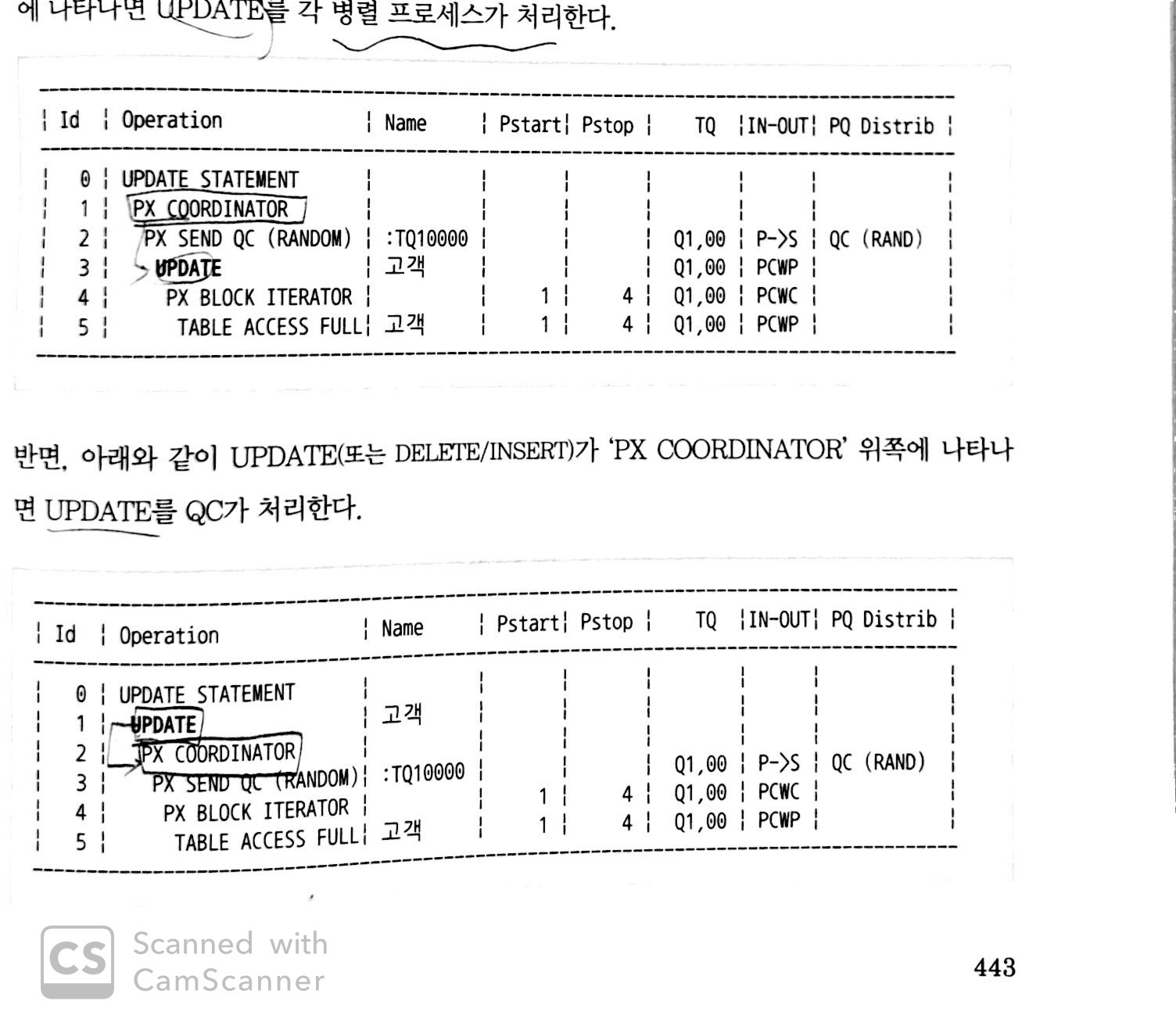
- 병렬 DML도 Direct Path Write 방식 사용하므로 Exclusive 모드 TM Lock 걸리는 것 주의!!

- 병렬 DML이 잘 작동하는지 확인하는 방법

-> DML 작업을 각 병렬 프로세스가 처리하는지 QC(조정쿼리)가 처리하는지 실행계획에서 확인



-> PX\_COORDINATOR 아래에 UPDATE가 있으면 UPDATE를 각 병렬 프로세스가 처리



-> PX\_COORDINATOR 위에 UPDATE가 있으면 UPDATE를 QC가 처리