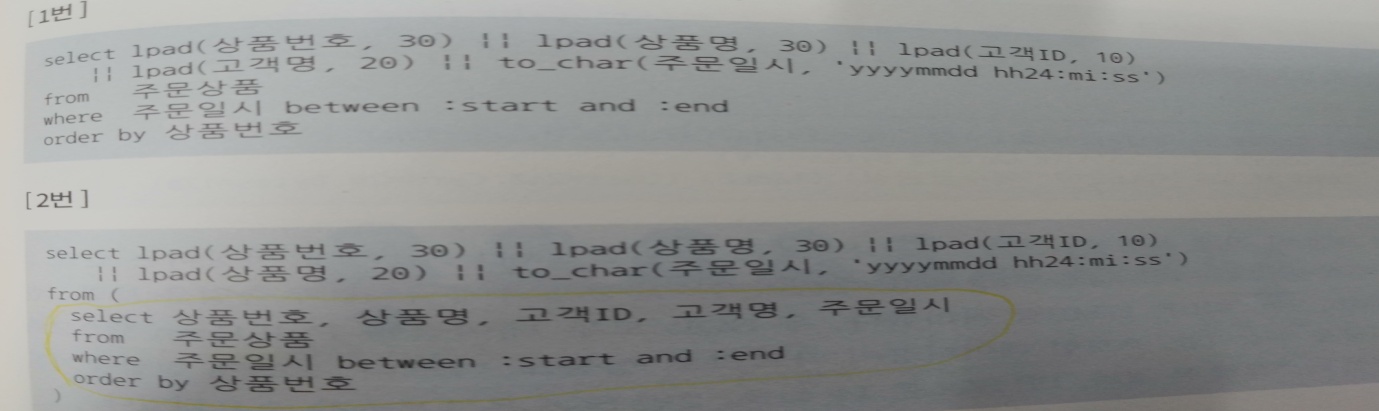
**5.4. Sort Area를 적게 사용하도록 SQL 작성**

- 소트 연산이 불가피하다면 메모리 내에서 처리를 완료할 수 있도록 Sort Area를 적게 사용할 수 있게 SQL 작성

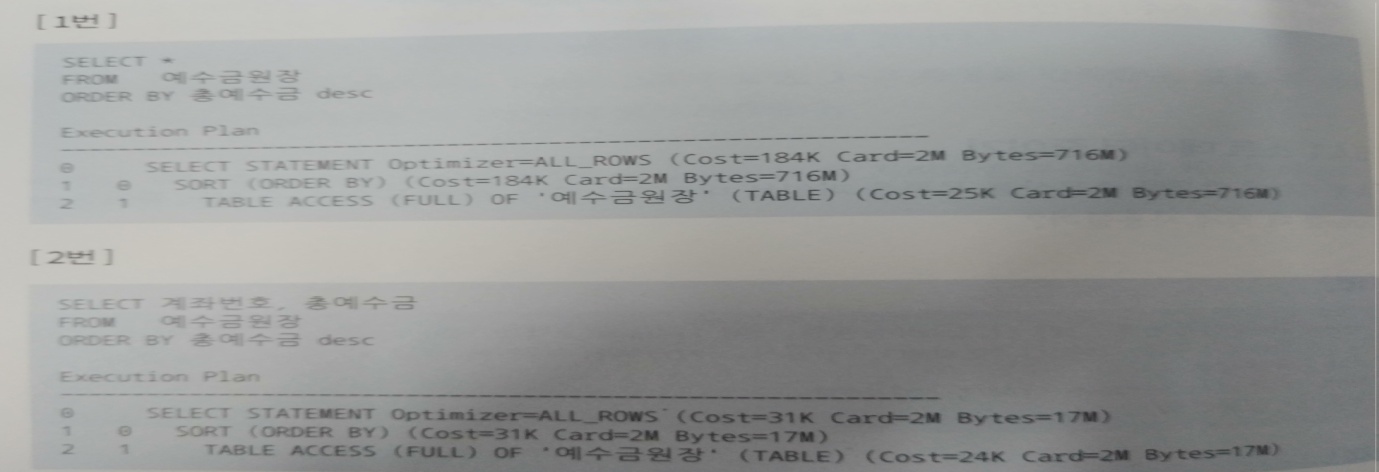
**5.4.1 소트 데이터 줄이기**

『예제 1. 특정 기간에 발생한 주문상품 목록 산출 SQL Sort Area 사용 비교』



- 1번의 경우 레코드당 가공한 결과집합을 Sort Area에 담는 반면, 2번의 경우 가공하지 않은 상태로 정렬을 완료하고 그 결과집합을 Sort Area 저장 후 최종 출력할 때 가공(Sort Area 에 저장되어 있는 데이터). **=> 2번 SQL이 Sort Area를 훨씬 적게 사용.**

『예제 2. 조회 대상 컬럼의 수에 따른 SQL Sort Area 사용량 비교』



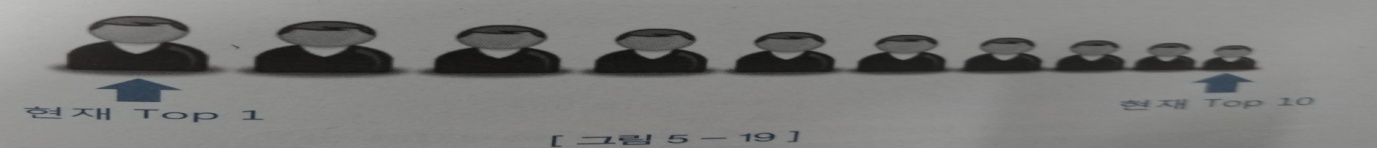
- 1번 SQL은 모든 컬럼을 Sort Area에 저장하는 반면, 2번 SQL은 계좌번호와 총예수금만 저장하기 때문에 2번 SQL이 적게 사용, 성능 차이 발생(예제 [예수금원장] 테이블이 229개의 컬럼을 가지고 있다고 가정할 경우, 1번 SQL은 14.1초, 2번 SQL은 1.2초 소요 & 716M vs. 17M).

**=> 조회 대상 컬럼이 적을수록 SQL Sort Area 에 저장할 데이터가 줄어들고 성능이 개선됨.**

**5.4.2 Top N 쿼리의 소트 부하 경감 원리**

- 인덱스로 소트 연산을 생략할 수 없을 때(Top N Stopkey 알고리즘 사용 불가 케이스), Top N 쿼리 동작 원리: Top N 소트 알고리즘.

『Top N 소트 알고리즘 동작원리 - 전교생 1,000명 중 가장 키 큰 학생 열 명 선발』 - p383



1. 대상 TABLE FULL SCAN.
2. Top N 표본 데이터 임의 산출 및 선별 기준에 따른 정렬, 그리고 Sort Area 저장.
3. 마지막 N 번째 데이터와 남은 데이터들을 선별 기준에 따라 비교, 남은 데이터들 중에서 선별 조건에 보다 부합하는 데이터가 있는 경우 해당 데이터를 기존 N 번째 자리 대체.
4. Sort Area Top N 데이터 정렬.
5. 남은 ① 번 TABLE FULL SCAN 데이터로 ③ 번, ④ 번 과정 반복.

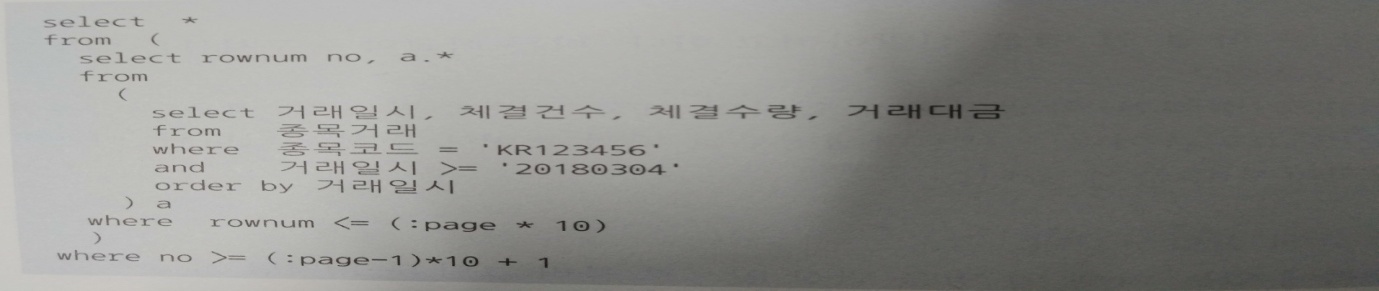
■ Sort Order By 오퍼레이션: SORT ORDER BY STOPKEY

- Sort Order By 옆에 보이는 ‘Stopkey’ 키워드(‘SORT ORDER BY STOPKEY’)는 해당 SQL 실행 과정에서 소트 연산을 피할 수 없어 Sort Order by 오퍼레이션을 수행하지만 ‘Top N 소트’ 알고리즘이 작동한다는 것을 실행계획에 표시하는 것임.

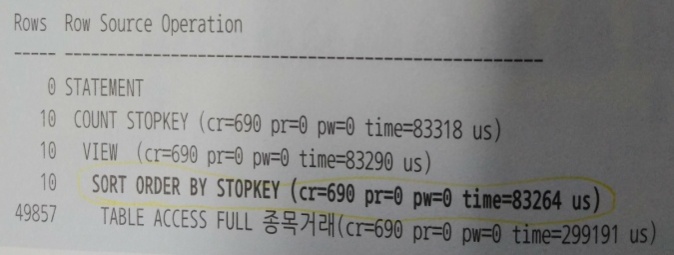
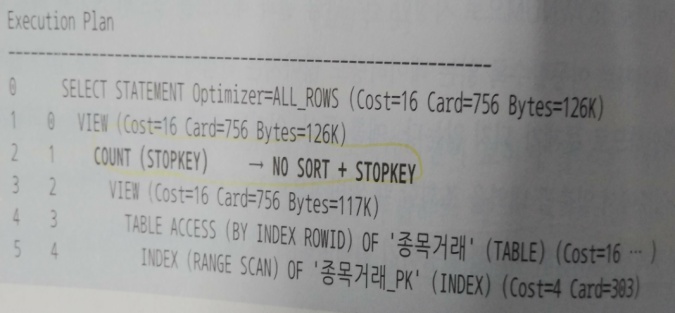
- ‘Top N 소트’ 알고리즘이 작동: 소트 연산(=값 비교) 횟수와 Sort Area 사용량을 최소화.

- DISK상 Physical Read(=pr)와 Physical Write(=pw)가 발생하지 않음 – AutoTrace 확인.

『예제 3. 인덱스 소트 연산 vs. Table Full Scan 방식에 따른 Top N 쿼리 동작 방식 비교』



- 예제3. 인덱스 소트 연산. - 예제3. Table Full Scan 방식.

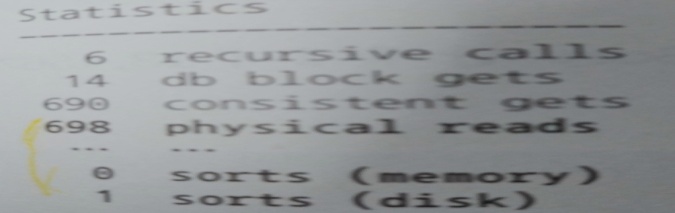
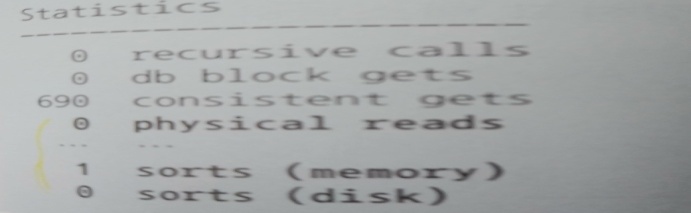


**5.4.3 Top N 쿼리가 아닐 때 발생하는 소트 부하**

- Order By 아래 ROWNUM 조건절 제거: p386 SQL & p.387 실행계획(STOPKEY 없음) 참조.

『AutoTrace 통해 살펴본 Top N 쿼리 적용 여부에 따른 성능 비교(p.385 vs. p387 참조)』

- Top N 쿼리 적용: sorted in momory area - Top N 쿼리 적용 불가: sorted in disk area

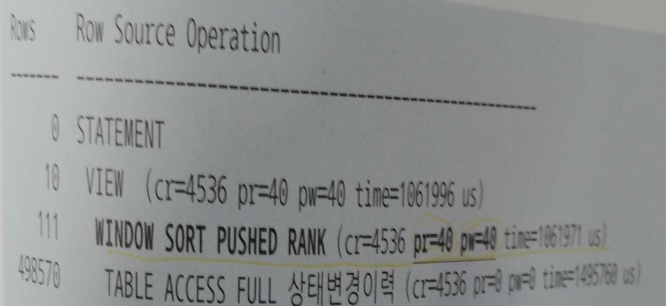
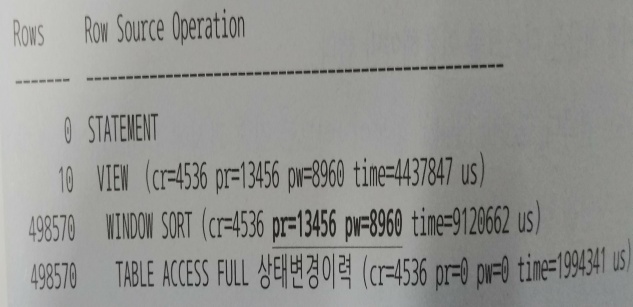


**5.4.4 분석함수에서의 Top N 소트**

- 윈도우 함수 중 Top N 소트 알고리즘이 작동할 경우 소트 부하가 적음

: **‘rank()’ or ‘row\_number()’ > ‘max()’** – rank(), row\_number() 의 경우 max() 함수와 다르게 Top N 소트 알고리즘이 작동됨.

『예제.4 상태변경이력 max vs. rank 사용 성능 비교 – p.388 참조』



※ 왼쪽: ‘max(변경순번) over (partition by 장비번호) 최종변경순번’ 적용

※ 오른쪽: ‘rank() over (partition by 장비번호 order by 변경순번 desc) rnum’ 적용