Oracle 11xe 환경에서의 대용량 데이터 (기초) **페이징**

Problem :

실제 서비스 환경을 가정하고 , 한계 테스트를 위해
테이블에 **200** 만개 이상의 데이터가 스크립트로 **insert** 되어
페이지 로딩 속도가 현저히 느려짐 .

Paging Query

- Oracle 12c 이상부터 ANSI 표준 페이징 문법인 OFFSET … FETCH 가 도입됨 .
- Oracle 11g XE 에서는 ROWNUM + 서브쿼리의 전통적 방식으로 페이징 해야함 .

```
SELECT * FROM
    (SELECT rownum as rnum, n.* FROM
    (SELECT board_no, board_title, board_writer, board_date
    FROM board ORDER BY board_no DESC )n )
WHERE rnum BETWEEN #{start} and #{end}
```

rownum 이 필요한 이유 → board_no 는 데이터의 연속성이 손상되어 있을 수 있음
→ 페이징 깨짐
rownum 은 데이터를 읽는 순서대로 번호를 부여 (where 보다 먼저 실행됨, 정렬전에 부여)

내부부터 인라인뷰 → 서브쿼리 (rownum 부여) → 최외곽쿼리 (범위필터링)

< 최신글 정렬 >

```
(SELECT board_no, board_title, board_writer, board_date
FROM board ORDER BY board_no DESC )n )
```

<rownum 부여 >

```
(SELECT rownum as rnum, n.* FROM (Inline View)n)
```

< 범위 지정 >

```
SELECT * FROM
( 내부 )
WHERE rnum BETWEEN #{start} and #{end}
```

Offset 기반의 query 에서 주목할 점

1. 대용량 데이터일 수록 프로세스가 비효율적

   (1) 전체 테이블 스캔 → 정렬
   (2) 정렬된 결과에 **rownum** 부여
   (3) **BETWEEN X and Y → X** 에서 **Y** 까지만 선택,
   그 이전 결과는 버림.

2. 대안

   (1) 인덱스 활용 (특정번호 이후 조회 → **Mark**)
   (2) 페이지 제한 (최대 페이지) → 일정량 이상의 데이터는 페이지 분할
   (3) 검색조건 제한 (예 : 일정 시점부터 일정 기간까지)

Advanced

- Cursor 기반 페이징
- Keyset 기반 페이징
- 하이브리드 접근
- 파티셔닝

→ 쿼리에 따른 백엔드 피드백과 UI 수정이 불가피