3.2 | 부분범위 처리 활용

목적

: 테이블 랜덤 액세스로 인한 인덱스 손익분기점의 한계를 극복하기 위한 것

: 빠른 응답속도

부분범위 처리

: DBMS가 클라이언트에게 데이터를 전송할 때 일정량씩 나누어 전송.

: 전송 후 데이터가 남아있어도 클라이언트가 추가 **Fetch Call 할 때까지 대기**

: DBMS가 먼저 읽는 데이터부터 Array Size(일정량)을 전송하고 멈춘다.

: 데이터 전송 후 서버 프로세스는 CPU를 OS에 반환하고 대기 큐에서 잠을 잔다.

* **연속 전송이 아닌 사용자의 Fetch Call이 있을 때마다 일정량씩 나누어 전송하는 것**

ArraySize가 10인 자바 프로그램의 데이터 읽기 메커니즘

1. 처음에 데이터 10건을 캐시에 저장
2. rs.next()시 캐시에서 데이터를 읽는다.
3. 캐시 데이터 소진시 rs.next() 호출 시 Fetch Call을 발생하여 데이터 10건 전송받는다.
4. 2-3번 반복

정렬 조건이 있을 때 부분범위 처리 – order by

: 모든 데이터를 읽어 정렬 후에 데이터 전송 = 전체범위처리

: **created 컬럼이 선두인 인덱스**가 있는 경우 항상 정렬된 상태이므로 결과집합 바로 전송 가능 = 부분범위 처리

: Array Size는 클라이언트 프로그램에서 설정가능 . – [3-16그림] 162p 참고

대량 데이터일 경우 Array Size를 조절(늘려)하여 Fetch Call 횟수 감소

부분처리 원리 활용하는 툴 – 오렌지, 토드

부분범위 처리 구현 - 165p JAVA 코드 //보통은 개발 프레임워크에 미리 구현된 기능을 사용

OLTP 부분범위 처리에 의한 성능개선 원리

OLTP – 온라인 트랜잭션을 처리하는 시스템, 소량의 데이터를 읽고 갱신

167p **인덱스 선두컬럼에 의해 sort 연산 생략**

\*\*앞쪽 일부만 출력 후 멈춘다. 대부분의 툴이 이미 구현되어 있다.

(배치 I/O

* 블록 건당 I/O call을 발생시키는 비효율을 줄이기 위한 기능. 테이블 블록 디스크 I/O Call을 미뤘다가 한꺼번에 처리한다.
* 실제 배치 I/O 작동 시 데이터 출력 순서가 인덱스 정렬 순서와 다를 수 있다. = sort 생략 불가
* order by 절이 없는 경우에는 정렬 순서를 보장할 필요가 없어서 sort 연산을 안함 )