정렬은 SQL의 성능을 저하시킨다

SQL을 작성하다 보면 업무 요건에 의해 많은 곳에서 정렬을 수행하게 된다. 많은 SQL이 ORDER BY 절에 의한 정렬이다. 이와 같은 정렬이 SQL의 성능을 저하시킨다는 것은 누구나 다 아는 일일 것이다. 이제는 많은 부하를 발생시키는 정렬을 제거해 데이터베이스 시스템의 성능을 최적화할 시기이다. 정렬이 성능 저하를 발생시킨다는 것은 모두 알고 있을 것이다. 하지만 정렬이 어떤 이유에서 성능 저하를 발생시키고, 또한 어떤 경우에 정렬이 발생하는지 모르는 경우가 많다. 이러한 정확한 내용을 이해하지 못하기 때문에 정렬을 제거하는 방법도 잘 이해하지 못하는 것 같다. 이제부터 정렬이 왜 성능 저하를 발생시키는지와 정렬이 발생하는 이유를 확인해 보자. 정렬의 제거 또한 확인해 SQL 성능을 극대화하는 방법을 확인해 보자.

정렬은 언제 발생하는가?

어떤 경우에 정렬이 발생하는가? 일반적으로 우리가 알고 있는 경우를 생각해 보자. 대부분이 ORDER BY 절을 사용하면 정렬이 발생한다는 것을 이해하고 있을 것이다. 물론, DISTINCT를 사용해도 정렬이 발생한다는 것을 이해하고 있다. 그렇다면 이와 같은 경우에만 정렬이 발생하는 것일까? 정렬은 우리가 생각하지 않은 부분에서도 많이 발생하게 된다. 다음과 같은 경우 정렬이 발생해 성능 저하를 발생시킬 수 있다.

- ORDER BY 절
- DISTINCT
- 소트 머지 조인
- 해시 조인
- 집합 연산자(UNION ALL을 제외한 모든 집합 연산자)
- 분석 함수
- 그룹 함수(오라클 10g에서는 발생하지 않을 수 있음)
- IN 절의 사용(수행 방식에 따라 발생할 수 있음)
- 인덱스 생성

위와 같은 경우에 정렬이 발생한다. 이중 우리가 의도해서 정렬을 발생시키는 경우는 얼마나 되는가? ORDER BY 절과 DISTINCT를 제외하고는 사실 정렬을 의도한 것은 아니었지만 SQL을 수행하기 위해 데이터베이스 내부에서는 정렬 아키텍처를 사용한 것이다. 이와 같은 경우에 정렬이 발생할 수 있으며 이로 인해 성능은 저하될 수 있다.

정렬이 발생하는 이유를 정확히 이해하자?

위에서 정렬을 발생시키는 경우에 대해 몇 가지를 확인해 보자. 첫 번째로 소트 머지 조인과 해시 조인을 확인해 보자. 2개의 경우 조인 방식(JOIN METHOD)에 해당한다. 오라클에서 제공하는 조인 방식에는 다음과 같이 세 가지가 존재한다.

- 중첩 루프 조인(NESTED-LOOPS JOIN)
- 소트 머지 조인(SORT MERGE JOIN)
- 해시 조인(HASH JOIN)

위와 같이 세 가지 조인 방식이 존재하며 조인 방식은 조인을 해결하기 위해 데이터베이스 내부에서 사용하는 방식을 의미한다. 결국, 2개 이상의 테이블에서 원하는 데이터를 추출하기 위해서는 위의 세 가지 방식 중 하나의 방식을 이용해야 한다. 이와 같은 세 가지 방식 중 소트 머지 조인과 해시 조인은 정렬을 수행하게 되므로 정렬에 의한 성능 저하가 발생할 수 있다. 조인 방식에 의해 발생하는 정렬은 우리가 의도하지 않은 정렬의 대표적인 예일 것이다.

두 번째로 집합 연산자의 정렬을 확인해 보자. 집합 연산자는 많은 종류가 존재한다. 그 중 UNION ALL을 제외한 모든 집합

연산자는 내부적으로 정렬을 수행하게 된다. 집합 연산자가 정렬을 수행하는 이유는 매우 간단하다. 2개의 집합에서 교집합을 추출하고자 한다면 정렬을 수행해야 한다. 교집합을 추출하지 않더라도 교집합의 모양을 확인하고자 한다면 반드시 정렬을 이용해야만 가능하다. 이와 같은 이유에서 대부분의 집합 연산자는 정렬을 이용하게 된다.

세 번째로 그룹 함수의 정렬을 확인해 보자. 그룹 함수의 정렬은 GROUP BY 절을 사용하는 경우 발생하게 된다. 하지만, 오라클의 경우 버전이 높아지면서 이와 같은 GROUP BY 절에 의한 정렬보다는 해시 함수를 이용하게 되었다. 그렇기 때문에 지금의 버전에서는 그룹 함수에 의한 정렬은 거의 발생하지 않지만 아직 파라미터 등을 이용해 기존의 정렬 방식으로 GROUP BY를 수행하는 경우도 많다. GROUP BY는 동일한 데이터를 하나의 값으로 추출하게 된다. 이와 같은 이유에서 정렬을 수행하게 되는 것이다.

네 번째로 분석 함수를 확인해 보자. 분석 함수는 무엇인가? 분석 함수는 일반 그룹 함수보다는 한 단계 진보된 형태이다. 다음의 예제를 확인해 보자.

```
SQL> SELECT ROW_NYMBER() OVER(PARTITION BY 1 ORDER BY ARR_DATE) 순번,
          ......
     FROM TN_BOARD
     WHERE  BOARD_ID =  '111'
```

위와 같이 SQL을 수행한다면 WINDOW(SORT)라는 실행 계획이 생성된다. WINDOW(SORT) 실행 계획은 무엇인가? 예제에서는 ROW_NUMBER 분석함수를 사용했다. 분석 함수는 해당 함수의 옆에 OVER 절을 설정하게 된다. OVER 절은 무엇인가? OVER 절은 데이터를 논리적으로 분할하는 PARTITION BY 절과 해당 논리적인 분할 영역을 정렬하는 ORDER BY 절로 구분된다. 이 중에서 ORDER BY 절은 각각의 논리적인 PARTITION을 정렬하게 되며 이와 같은 논리적인 PARTITION을 WINDOW라고 부르게 된다. 따라서 실행 계획에는 WINDOW(SORT)라는 실행 계획이 생성되며 이와 같은 분석 함수들도 결국 정렬을 수행하게 된다.

다섯 번째로 IN 절에서 발생하는 정렬을 확인해 보자. 다음의 예제를 확인해 보자.

```
SQL> SELECT 거래번호, 카드번호, 사용자_이름,
FROM 거래내역
     WHERE 카드번호 IN (SELECT 카드번호 FROM 카드
                WHERE 가입일 >= '20090901' )
```

위의 SQL은 어떠한가? 위의 SQL의 수행 방법을 확인해 본다면 두 가지 방식 중 하나가 발생하게 된다. 카드 테이블이 먼저 엑세스되는 경우와 거래내역 테이블이 먼저 엑세스되는 두 가지 경우가 존재한다. 이 중 카드 테이블이 먼저 엑세스되는 경우는 어떠한가? 이와 같이 카드 테이블이 먼저 엑세스되는 경우에 정렬이 발생하게 된다. SORT(UNIQUE)가 발생하게 되며 이는 유일한 값을 추출하기 위한 정렬을 의미하게 된다. 왜 이와 같은 현상이 발생하게 되는가? 다음의 예제를 확인해 보자.

```
SQL> SELECT 거래번호, 카드번호, 사용자_이름,
FROM 거래내역
     WHERE 카드번호 IN ( '111' ,' 112' , '113' , '111' )
```

위의 SQL에서 '111' 번 카드번호는 IN 절에 두 번 존재하지만 이는 한 번만 수행되게 된다. 즉, IN 절의 상수를 유일한 값으로 정렬해 한 번씩만 수행해 결과 값을 추출하기 때문이다. 이와 같이 IN 절의 테이블이 먼저 엑세스되는 경우 정렬이 발생하게 된다.

정렬을 발생시키는 경우는 다양하게 존재한다. 적을 알고 나를 알면 100전 100승이라고 했듯이 정렬이 어떤 경우에 발생하는지를 정확히 이해해야만 우리는 정렬을 제거해 성능을 최적화할 수 있다. 정렬을 이해하고 제거하는 것은 데이터베이스 시스템의 성능 향상을 위해 매우 중요한 요소임을 기억하길 바란다.

권순용 kwontra@hanmail.net | Data Consulting 업무를 수행하는 ㈜엑시엄의 대표이사이며 DBA로 시작해 SQL 튜닝, 데이터베이스 아키텍처 및 모델링 업무를 주로 수행했다. 데이터베이스 교육에도 많은 관심을 가지고 있으며 저서로는 『Perfect! 오라클 실전 튜닝』, 『초보자를 위한 오라클 10g』 및 『INSIDE SQL』이 있다. 또한, 데이터 액세스 최적화에 대한 특허를 출원했다.