복잡한 SQL도 정렬을 제거할 수 있다

SQL에서 정렬을 제거한다는 것은 무엇을 의미하는가? SQL에서 정렬을 제거하는 것은 특이한 SQL을 제외하고는 대부분이 가능하다. 그렇다고 모든 SQL에 대한 정렬을 제거할 필요는 없다. 해당 데이터베이스의 모든 SQL에 대한 정렬을 제거하고자 하는 것은 어떻게 보면 매우 무모한 도전일 것이다. 정렬을 수행하는 SQL 중 문제 SQL 또는 매우 자주 수행되는 SQL에 대해 정렬을 제거하고자 한다면 해당 데이터베이스는 지금보다 훨씬 최적화될 것이다.

조인이 사용되어도 정렬을 제거할 수 있다

조인을 사용하는 경우에도 인덱스를 이용해 원하는 컬럼으로 자동 정렬을 수행할 수 있다. 물론, 어려운 정렬 SQL의 경우에도 함수 기반 인덱스를 사용해 원하는 컬럼으로 정렬을 수행할 수 있다.

```
SQL> SELECT A.사원이름, A.입사일자, B.부서이름, B.부서 위치
     FROM 사원 A, 부서 B
     WHERE A.사원번호 = B.사원번호
     AND A.지역 = '서울'
     ORDER BY 사원이름;
```

위와 같은 조인 SQL이 존재한다고 가정하자. 그렇다면 어떻게 우리는 ORDER BY 절을 사용하지 않고 해당 SQL을 정렬할 수 있겠는가? ORDER BY 사원이름 절을 제거해도 과연 정렬된 데이터가 추출될 수 있는가? 정답은 'ORDER BY 절을 제거하고 자동으로 정렬된 데이터를 추출할 수 있다' 이다. 그렇다면 위와 같은 조인 SQL은 어떻게 정렬된 데이터를 자동으로 추출할 수 있겠는가? 이를 이해하기 위해서는 데이터베이스에서 제공하는 조인 방식(JOIN METHOD)을 이해해야 할 것이다. 오라클의 경우 제공되는 조인 방식은 다음과 같이 세 가지가 존재한다.

- NESTED-LOOPS JOIN (인덱스를 이용한 정렬 제거)
- HASH JOIN
- SORT MERGE JOIN

앞의 SQL이 NESTED-LOOPS 조인으로 수행되었다고 가정하자. 그렇다면 위의 SQL은 어떻게 수행되겠는가? 조인은 2개 이상의 테이블에서 원하는 컬럼을 추출하는 SQL이다. 그렇기 때문에 FROM 절에 2개 이상의 테이블이 사용되는 것은 너무나 자명하다. FROM 절에 존재하는 모든 테이블을 액세스해야 원하는 결과를 추출할 수 있으므로 조인은 FROM 절에 존재하는 모든 테이블을 액세스하게 된다. 이와 같은 경우 한 번에 모든 테이블을 액세스할 수는 없으므로 먼저 하나의 테이블을 액세스하고 그 다음에 다른 테이블을 액세스하는 방식을 이용하게 된다. 이를 조인 순서라고 한다. 결국, 조인이 수행된다는 것은 어떤 테이블은 먼저 액세스되고 어떤 테이블은 뒤에 액세스되어 원하는 결과를 추출하게 되는 것이다. NESTED-LOOPS 조인의 경우에는 먼저 액세스되는 테이블을 DRIVING 테이블이라 하며 뒤에 액세스되는 테이블을 INNER 테이블이라고 부른다. 그렇다면 위의 예제에서 사원 테이블이 DRIVING 테이블이며 부서 테이블이 INNER 테이블이라고 가정하자. 그렇다면 해당 SQL은 어떻게 수행되겠는가? 사원 테이블의 조건은 지역 컬럼의 값이 '서울' 인 데이터이다. 그렇기 때문에 지역 컬럼의 값을 만족하는 데이터만을 먼저 액세스한 후 해당 데이터로 사원 테이블을 액세스하게 된다. 지역 컬럼의 조건을 만족하는 데이터가 100건이라면 한 건씩 부서 테이블과 조인을 수행해 결과를 추출하게 된다. 바로 이점을 유의하자. 사원 테이블에서 추출되는 데이터가 사원이름 순으로 추출된다면 부서 테이블과의 조인도 사원 테이블에서 추출되는 데이터의 순서대로 결과가 추출되므로 최종 결과는 사원이름으로 정렬된 데이터가 자동으로 추출된다. 이를 정리하면 아래와 같다.

- NESTED-LOOPS JOIN : DRIVING 테이블에서 추출되는 데이터의 순서대로 최종 결과 추출

이와 같기 때문에 사원 테이블을 액세스해 추출되는 데이터의 순서에 의해 결과는 추출된다. 그렇기 때문에 사원 테이블에서 액세스하는 데이터가 사원이름의 순으로 데이터를 추출할 수 있으면 될 것이다. 사원 테이블에 대한 액세스를 확인해 보자.

```
SQL> SELECT A.사원이름, A.입사일자
     FROM 사원 A
     WHERE A.지역 = '서울'
     ORDER BY 사원이름;
```

사원 테이블은 위와 같이 지역 컬럼의 값이 서울인 데이터를 추출하는 SQL에 해당한다. 그렇다면 위의 SQL은 어떻게 자동으로 정렬된 데이터를 추출할 수 있겠는가? 이는 매우 간단하다. 사원 테이블에 인덱스를 지역+사원이름으로 생성한다면 해당 SQL은 자동으로 정렬된 데이터가 추출될 것이다. 지역+사원이름 인덱스는 지역 컬럼으로 정렬되어 있고 동일한 지역 컬럼에 대해서는 사원이름 컬럼으로 정렬되어 있으므로 인덱스에서 지역 컬럼의 값이 '서울'인 데이터를 추출한다면 결과는 사원이름 컬럼으로 정렬된 데이터가 추출될 것이다. 따라서 앞의 SQL이 자동으로 정렬된 데이터가 추출되기 위해서는 다음과 같은 방식으로 수행되어야 할 것이다.

- 조인 순서 : 사원 테이블이 DRIVING 테이블로 수행되며 부서 테이블이 INNER 테이블로 수행되어야 함
- 조인 방식 : NESTED-LOOPS JOIN을 이용해야 함
- 사용 인덱스 : 사원 테이블은 반드시 지역+사원이름 인덱스를 이용해야 함

SQL의 정렬 제거를 반드시 수행해야 하는가?

우리가 많이 사용하는 목록 쿼리에 대해 예를 들어보자. 목록 쿼리는 무엇인가? 목록 쿼리는 웹 환경에서 많이 사용하게 된다. 조건을 만족하는 데이터를 모두 보여주는 것이 아니라 보통의 경우 첫 페이지에 30건 정도를 추출해 주고 밑에는 페이지 숫자를 제공해 원하는 페이지로 이동할 수 있게 하는 형식이다. 또는 처음 화면에 30건의 데이터를 추출하고 스크롤로 내리게 되면 나머지 데이터를 추출하는 형식이다. 목록 쿼리의 작성은 어떻게 하는가? 대부분 목록 쿼리는 정렬을 요구하기 때문에 다음과 같이 ORDER BY 절을 사용해 작성하게 된다.

```
SQL> SELECT 사원이름, 입사일자
FROM (SELECT ROWNUM RNUM, 사원이름, 입사일자
FROM (SELECT A.사원이름, A.입사일자
                  FROM 사원 A
                  WHERE A.지역 = '서울'
                  ORDER BY 사원이름
                 )
        )
      WHERE RNUM BETWEEN 1 AND 30
```

위의 예제는 한 페이지에 30건의 데이터를 추출하는 예제이다. 여기서 문제는 무엇인가? 조건을 만족하는 데이터가 10,000건이라고 가정하자. 그렇다면 가장 안쪽의 인라인 뷰에 의해 반드시 10,000건의 데이터를 액세스해야 한다. 이는 화면에 보이는 데이터는 30건에 불과하지만 해당 SQL은 10,000건의 데이터를 액세스해야 하므로 성능을 보장받을 수 없다는 것이다. 그렇다면 가장 안쪽의 인라인 뷰는 반드시 10,000건의 데이터를 액세스해야 하는 것인가? 이는 ORDER BY 절이 존재하는 한 피할 수 없는 부분이다. 모든 데이터를 액세스하지 않고 어떻게 정렬을 수행할 수 있겠는가? 결국, 위의 SQL을 최적화하기 위해서는 인덱스를 이용해 자동 정렬을 수행해야 할 것이다.

```
SQL> SELECT 사원이름, 입사일자
FROM (SELECT ROWNUM RNUM, 사원이름, 입사일자
           FROM 사원 A
WHERE A.지역 = '서울'
AND ROWNUM <= 30
          )
      WHERE RNUM BETWEEN 1 AND 30
```

위와 같이 SQL을 작성하기 위해서는 사원 테이블에는 반드시 지역+사원이름 인덱스가 존재해야 하며 해당 SQL은 실행 계획에서 지역+사원이름 인덱스를 이용해 결과를 추출해야 한다. 이와 같이 수행된다면 인라인 뷰는 자동으로 정렬된 데이터가 추출될 것이다. 자동으로 추출되는 데이터에 대해 1 페이지는 ROWNUM 연산자에 의해 30건의 데이터만 추출하면 해당 데이터는 1 페이지를 구성하는 데이터가 된다. 더 이상의 데이터를 추출할 필요는 없을 것이다. 하지만 2 페이지를 액세스하는 경우에는 ROWNUM 연산자의 조건이 '60'이 되면 주 쿼리에서 RNUM 값이 '31'부터 '60'까지를 추출하는 형식으로 변경되어야 할 것이다. 이와 같이 수행한다면 우리는 최소의 데이터를 액세스해 해당 목록 쿼리를 최적화할 수 있을 것이다.

목록 쿼리의 최적화는 많은 시스템에서 필수 요소이다. 목록 쿼리의 최적화는 정렬의 제거에 있으며 이를 실패한다면 SQL 최적화를 성공할 수 없게 된다. 또한, 목록 쿼리 최적화의 핵심은 정렬을 제거해 각 페이지에서 최소의 데이터를 액세스해야 한다는 것에 있음을 명심하길 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.