지금 우리에게 필요한 건 정렬의 제거이다

업무 요건에 의해 많은 곳에서 SQL에 정렬을 수행하게 된다. 대부분의 SQL이 ORDER BY 절에 의한 정렬이다. 이와 같은 정렬이 SQL의 성능을 저하시킨다는 것은 누구나 아는 사실일 것이다. 하지만, 개발에 시간이 없어서인지 아니면 정확한 지식이 없어서인지 실제 업무에서 정렬을 제거하고자 하는 사람은 거의 없는 것 같다. 더욱 아쉬운 것은 성능을 향상시키고자 하는 튜너들 중 많은 튜너들이 정렬을 제거하는 방법을 모른다는 것이다.

정렬의 제거는 해당 시스템의 성능을 몇 단계 향상시킬 수 있는 방법이다. 그럼에도 불구하고 아직 많은 사이트에서는 이러한 것이 이뤄지지 않고 있다. 왜 그런 것일까? 개발에 식간이 없어서인가? 그것은 아닌 것 같다. 필자의 의견으로는 바빠서라기보다는 몰라서가 더 맞는 답인 것 같다. 이제 우리는 이와 같이 정렬을 제거할 수 있는 아키텍처를 이해해 최적의 성능을 보장 받아야 할 것이다. 이것이야말로 대용량 데이터베이스로 변하는 지금 우리에게 반드시 필요한 기술이다. 이번 호에서 SQL의 정렬을 제거하는 방법을 하나하나 알아보도록 하자.

정렬을 제거하기 위해 필요한 요소는 무엇인가?

정렬을 제거하기 위해 필요한 것은 무엇일까? 필자는 정렬을 제거하기 위해 필요한 요소는 세 가지라고 생각한다.

- 인덱스
- 기술력
- 논리적인 사고

정렬을 제거하기 위해서는 위와 같은 요소들이 융합되어야만 한다. 첫 번째로 인덱스에 대해 확인해 보자. ORDER BY 절을 가지고 있는 SQL에 대해 정렬을 제거하려면 어떻게 해야 하는가? ORDER BY 절을 제거하고 정렬을 제거하기 위해서는 이미 정렬된 데이터가 존재해야 할 것이다. 정렬된 데이터가 존재하지 않는데 자동으로 정렬된 데이터를 추출할 수는 없다. 그렇다면 어디에 정렬된 데이터가 존재하는가? 테이블을 먼저 확인해 보자. 테이블에는 데이터가 저장되는 순서대로 저장되므로 실제 정렬된 데이터가 존재할 수 없게 된다. 만약 테이블에 원하는 형태의 정렬된 데이터가 존재한다면 이는 운이 좋은 경우이다. 이와 같기 때문에 테이블에서 정렬된 데이터를 원할 수는 없게 된다. 그렇다면 또 무엇을 확인해야 하는가? 바로 인덱스이다. 인덱스는 어떠한가? 인덱스는 인덱스를 구성하는 컬럼으로 정렬되어 있게 된다. 그렇기 때문에 인덱스를 COL1 컬럼으로 생성한다면 해당 인덱스는 COL1 컬럼의 값으로 정렬되어 디스크에 저장된다. 이처럼 인덱스를 정확히 이해하고 효과적으로 사용해야만 우리는 정렬을 제거할 수 있다.

두 번째로 기술력에 대해 확인해 보자. 여기서 말하는 기술력은 데이터베이스 및 SQL에 대한 기술력이다. SQL을 작성하다 보면 조인을 많이 이용하게 된다. 이와 같은 조인은 2개 이상의 테이블에서 원하는 데이터를 추출하는 경우에 해당한다. 오히려 하나의 테이블에서 원하는 데이터를 추출하는 경우보다는 2개 이상의 테이블에서 원하는 데이터를 추출하는 경우가 더 많을 것이다. 이와 같은 경우에는 조인에 대한 성격을 정확히 이해해야만 정렬을 제거하고 정렬된 데이터를 추출할 수 있게 된다. 예를 들어 중첩 루프 조인(Nested Loops Join)의 경우에는 먼저 액세스되는 테이블이 이용하는 인덱스에 의해 정렬된 데이터가 추출될 것이다. 이처럼 인덱스만으로 정렬을 제거할 수 없으며 이에 따르는 전반적인 지식이 있어야만 우리는 정렬을 제거할 수 있다.

세 번째로 논리적인 사고이다. 여기서 말하는 논리적인 사고는 ORDER BY 절을 제거하기 위해 반드시 필요하다. ORDER BY 절을 제거하지 못하더라도 최적의 성능을 보장하기 위해 반드시 필요한 사항이다. 예를 들어 UNION ALL이 사용되었다면 어떻게 될까? 두 집합을 연결하는 연산자가 UNION ALL 집합

연산자이다. 이와 같은 경우 정렬을 수행해 해당 집합에서 20건의 데이터만을 추출한다면 쉽게 ORDER BY 절을 제거할 수 없을 것이다. 이런 경우에는 각각의 집합에서 정렬된 20건의 데이터를 추출한 후 40건에 대해 실제 ORDER BY 절을 수행해 정렬을 최적화할 수 있다. 이처럼 논리적인 사고는 ORDER BY 절을 제거하거나 최적화하기 위해 반드시 필요하다.

정렬을 제거한다는 것은 결코 쉬운 것이 아니다. 또한, 모든 정렬은 위에서 언급한 세 가지 요소를 정확히 구사한다면 모두 제거할 수 있다. 하지만 모든 정렬을 제거하는 것 또한 의미는 없으며 힘든 작업이 될 것이다. 그렇기 때문에 해당 시스템에서 중요한 SQL에 대해서만 정렬을 제거하는 것이 필요하다.

기본적인 정렬을 제거해 보자

우선적으로 기본적인 정렬을 제거해 봄으로써 정렬을 제거하는 기본 개념을 이해하길 바란다.

```
SQL> SELECT ……
     FROM TN_BOARD
     WHERE BOARD_ID = '111'
     ORDER BY REG_DATE;
```

위의 SQL에서 사용된 ORDER BY 절을 제거하기 위해서는 어떻게 해야 할까? 앞서 언급했듯이 ORDER BY 절을 제거하기 위해서는 인덱스를 이용해야 할 것이다. 그렇다면 어떻게 인덱스를 생성해서 이용해야 정렬을 제거하고 효과적인 SQL을 작성할 수 있겠는가?

REG_DATE로 인덱스를 생성했다고 가정하자. 그렇다면 인덱스에 BOARD_ID 컬럼이 존재하지 않으므로 모든 인덱스 값을 액세스해야 할 것이다. 모든 인덱스 값을 액세스한 후 BOARD_ID 값이 '111'인 데이터를 확인해 결과로 추출하게 된다. 이와 같이 수행한다면 인덱스 FULL SCAN 실행 계획이 생성될 것은 너무나도 자명한 일이다. 인덱스 FULL SCAN이 발생한다면 인덱스의 모든 값을 액세스한 후 BOARD_ID 컬럼의 값을 확인해 조건을 만족하지 않는 경우 버리게 되므로 비효율이 발생하게 되며 그 양이 많다면 성능 저하도 당연할 것이다. 인덱스를 BOARD_ID+REG_DATE로 생성하는 경우는 어떠한가? 이와 같이 인덱스를 생성한다면 BOARD_ID 컬럼을 만족하는 데이터에 대해 차례대로 값을 액세스하게 된다. 인덱스가 BOARD_ID+REG_DATE로 구성되어 있으므로 동일한 BOARD_ID 값에 대해서는 REG_DATE 컬럼의 값으로 정렬되어 있다. 그러므로 이와 같이 인덱스를 생성한다면 처리 범위도 감소시키면서 인덱스를 이용해 정렬을 제거할 수 있게 된다. 아래의 예제를 확인해 보자.

```
SQL> SELECT ……
     FROM TN_BOARD
     WHERE BOARD_ID BETWEEN '111' AND '200'
     ORDER BY REG_DATE;
```

위와 같이 SQL을 수행한다면 BOARD_ID+REG_DATE 인덱스로 정렬을 제거할 수 있겠는가? 해당 SQL은 BOARD_ID+REG_DATE 인덱스를 이용해서는 정렬된 데이터를 추출할 수 없다. 이는 BOARD_ID 컬럼의 값이 하나가 아니며 여러 개의 값이기 때문이다. BOARD_ID+REG_DATE 인덱스는 동일한 BOARD_ID 컬럼의 값에 대해서는 REG_DATE 컬럼의 값으로 정렬된다. 하지만 해당 SQL은 BOARD_ID 컬럼의 값이 여러 개이므로 조건을 만족하는 BOARD_ID 컬럼의 값에 대해 REG_DATE 컬럼의 값으로 정렬되어 있다고 할 수 없다. 그렇기 때문에 위의 예제는 BOARD_ID+REG_DATE 인덱스를 이용해 정렬된 데이터를 추출할 수 없게 되며 인덱스를 이용해 정렬을 수행하고자 한다면 REG_DATE 인덱스 또는 REG_DATE+BOARD_ID 인덱스를 이용해야 할 것이다. 이와 같다면 BOARD_ID 컬럼에 의해 처리 범위가 감소하지 않게 된다. 이 경우에는 정렬을 제거하고 전체를 액세스하는 것이 유리한지 아니면 정렬은 수행하되 BOARD_ID 컬럼에 의한 처리 범위를 감소시키는 것이 유리한지를 고려해 선택해야 할 것이다.

정렬을 제거하는 가장 기본적인 예제들을 확인해 보았다. 해당 예제들은 인덱스만을 이용해 정렬을 제거하는 것이며 인덱스 구성만 최적화한다면 손쉽게 정렬을 제거할 수 있다. 이처럼 정렬을 제거하고자 한다면 정렬을 제거할 수는 있다. 하지만, 정렬을 제거할 경우의 상황을 정확히 파악해 최적의 경우를 선택해야 한다는 것이 더 중요하다. 다음 호에서는 정렬을 제거하는 SQL에 대해 복잡한 경우를 확인해 보도록 하자.

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.