VLDB의 성능을 고려한 데이터 모델링에 관한 연구

이 종 석* · 이 창 호**
*남서울대학교 산업경영공학과 · **인하대학교 산업공학과

A Study on Data Modeling for VLDB Performance

Zhong-Shi Li* · Chang-Ho Lee**
*Department of Industrial & Management Engineering, NAMSEOUL University
**Department of Industrial Engineering, INHA University

Abstract

It has been a huge amount of capacity of 10GB data base in a decade ago so far. Nowadays, however, 10TB is the common data base and even bigger capacities are available. So, new generation of Very Large Data Base (VLDB) has begun.

Moving in to the new generation of VLDB has been caused major problems like backing up, restoring, and managing especially performance. It is very hard to export necessary data rapidly now due to the huge amount of data base. In the past, such kind of problems was out of the questions because of less data. As time goes on, however, optimization of performance became a big issue when the VLDB is common. Therefore, new professional technics are urgently required to maintain and optimize the data base that has become a VLDB or one that is in the progress of becoming one.

Keywords: Data Modeling, VLDB, EPCglobal Network.

1. 서 론

데이터의 용량이 커질수록 기업의 의사결정의 속도가 빨라질수록 데이터를 처리하는 속도는 빠르게 처리되어야 할 필요성을 반증해 준다. 일반적으로 실무 프로젝트에서 보면 잘못된 테이블 디자인 위에서 개발된 어플리케이션의 성능이 저하되는 경우, 개발자가 구축한 SQL 구문에 대해서만 책망을 하는 경우가 많이 있다. 물론 개발자가 SQL 구문을 잘못 구성하여 성능이 저하되는 경우도 있지만 근본적으로 디자인이 잘못되어 SQL 구문을 잘못 작성하도록 구성될 수밖에 없는 경우도 빈번하게 발생되고 있다.[6][10][16][18][20]

성능이 저하되는 데이터 모델은 크게 세 가지 경우로 나눌 수 있다. 데이터 모델 구조에 의해 성능이 저하될 수도 있고 데이터가 대용량으로 됨으로 인해 불가피하게 성능이 저하되어 나타나는 경우도 있다. 또한 인덱스 특성을 충분히 고려하지 않고 인덱스를 생성함으로 인해 성능이 저하되어 나타나는 경우도 있다.[6][16][20]

본 논문에서는 대용량 데이터의 효율적인 관리, 처리 및 유지를 위해 데이터 모델의 성능을 향상시킬 수 있는 방안을 조사 연구하고자 한다.

2. 이론적 배경

2.1 VLDB

데이터는 살아 움직이고 있다. 10년 전만 해도 10GB 정도의 데이터라면 대용량 데이터라고 불리던 시절이 있었다. 하지만 지금은 10TB보다 큰 데이터베이스도 흔하다. 결국, 대용량 데이터베이스(Very Large DataBase, VLDB)의 시대가 개막된 것이다.

† 교신저자: 이종석, 남서울대학교 산업경영공학과
M · P : 010-2631-9780, E-mail: leezs0423@nsu.ac.kr
2012년 4월 9일 접수; 2012년 6월 7일 수정본 접수; 2012년 6월 7일 게재확정

VLDB로 변한 데이터베이스에는 백업, 복구, 관리와 같은 문제점이 있지만 그 중에서도 성능 문제를 빼놓을 수 없다. 데이터베이스에 많은 데이터가 있고 그렇게 많은 데이터 중에서 필요한 몇 건의 데이터만 추출하는 것이 쉬운 일이 아니다. 과거에는 데이터가 적었기 때문에 이러한 것이 큰 문제가 아니었지만 이제는 VLDB가 되면서 성능 최적화는 일상적이고도 중요한 이슈가 되었다. 따라서 VLDB가 된 데이터베이스나 VLDB로 변하고 있는 데이터베이스에서 성능 관리를 하고 최적화할 수 있는 전문 기술이 필요하다.

2.2 성능 데이터 모델링

성능 데이터 모델링이란 데이터베이스 성능 향상을 목적으로 설계단계의 데이터 모델링 때부터 정규화, 반정규화, 테이블통합, 테이블분할, 조인구조, PK, FK 등 여러 가지 성능과 관련된 사항이 데이터 모델링에 반영될 수 있도록 하는 것으로 정의할 수 있다.[6][16]

성능 데이터 모델링은 정규화를 통해서도 수행할 수 있고 인덱스의 특징을 고려해서 컬럼의 순서도 변형할 수 있다. 또한 대량의 데이터 특성에 따라 비록 정규화된 모델이라도 테이블을 수직 또는 수평 분할하여 적용하는 방법도 있고 논리적인 테이블을 물리적인 테이블로 전환할 때 데이터 처리의 성격에 따라 변환하는 방법도 성능 데이터 모델링의 범주에 포함될 수 있다.[16]

정규화된 모델이 데이터를 주요 관심사별로 분산시키는 효과가 있기 때문에 그 자체로 성능을 향상시키는 효과가 있다. 각각의 엔티티에 대한 용량산정을 수행하면 어떤 엔티티에 데이터가 집중되는지 파악할 수 있다. 또한 데이터 모델에 발생되는 트랜잭션의 유형을 파악할 필요가 있다. 트랜잭션의 유형을 파악하게 되면 SQL 문장의 조인관계 테이블에서 데이터 조회의 컬럼들을 파악할 수 있게 되어 그에 따라 성능을 고려한 데이터 모델을 설계할 수 있다.

이렇게 파악된 용량산정과 트랜잭션의 유형 데이터를 근거로 정확하게 테이블에 대해 반정규화를 적용하도록 한다. 반정규화는 테이블, 속성, 관계에 대해 포괄적인 반정규화의 방법을 적용해야 한다. 또한 대량 데이터가 처리되는 이력모델에 대해 성능 고려를 하고 PK/FK의 순서가 인덱스 특성에 따라 성능에 영향을 미치는 영향도가 크기 때문에 반드시 PK/FK를 성능이 우수한 순서대로 컬럼의 순서를 조정해야 한다.

2.3 정규화

정규화는 다양한 유형의 검사를 통해 데이터 모델을 좀 더 구조화하고 개선시켜 나가는 절차에 관련된 이론이다. 정규화의 기본 원칙은 하나의 테이블에는 중복된 데이터가 없도록 하는 것이다.[16][20]

정규화의 특징은 다음과 같다.[16]

1) 정규화는 적절한 엔티티 타입에 각각의 속성들을 배치하고 엔티티 타입을 충분히 도출해가는 단계적인 분석 방법이다.

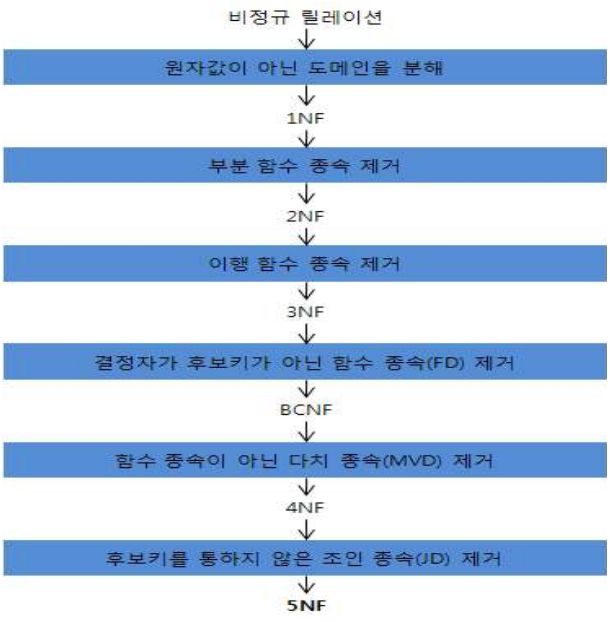
2) 정규화 기술은 엔티티 타입에 속성들이 상호 종속적인 관계를 갖는 것을 배경으로 종속 관계를 이용하여 엔티티 타입을 정제하는 방법이다.

3) 각각의 속성들이 데이터 모델에 포함될 수 있는 정규화의 원리를 이용하여 데이터를 분석하는 방법에서 활용될 수 있다.

4) 정규화는 현재 데이터를 검증할 수 있고 데이터의 표현 관점에서 엔티티 타입을 정의하는 데 이용할 수 있다.

5) 정규화는 엔티티 타입을 오브젝트 별로 분석하는 방법이 아닌 개별 데이터를 이용한 수학적인 접근 방법을 통해 분석한다.

현재 정규화 이론은 6단계까지 있다. 그러나 일반적으로 3단계까지만 사용한다.[11]



[그림 1] 정규화

2.4 반정규화

반정규화는 정규화된 엔티티 타입, 속성, 관계를 시스템의 성능 향상, 개발과 운영을 단순화하기 위해 데

이터 모델을 통합하는 프로세스를 말한다.[16][20]

대부분의 업무에서는 정확한 데이터의 관리가 중요한 관건이다. 그래서 데이터의 정합성과 무결성을 보장할 수 있는 정규화가 기본적인 전제라 할 수 있다.

그러나 테이블의 복잡성과 시스템의 성능을 고려하지 않을 수 없으므로 기본적으로는 정규화한 테이블을 그대로 유지하는 것을 목표로 하고, 문제가 되는 테이블에 대해서 뷰의 생성, 파티셔닝 테이블 생성, 인덱스 조정, 클러스터링 적용 등 여러 가지 방안을 먼저 조사하도록 한다. 그 다음 반정규화를 고려한다.

반정규형을 사용하는 유일한 목적은 조회 성능을 향상시키기 위해서이다. 데이터베이스의 조회 성능을 향상시키기 위함이 아니라면 반정규화는 잊어야 한다. 왜냐하면 반정규형은 어떤 식으로든 중복된 데이터를 관리해야 하는데, 중복된 데이터는 원 데이터와 정합성을 맞춰야 하기 때문이다.[5]

반정규화는 모델링을 수행할 때 반드시 필요한 기법 중의 하나이다. 남용하지 않고 적절히 사용하면 성능적으로 좋은 모델이 될 것이다. 반정규화를 할 때 반드시 고려해야 할 점은 정규화가 선행돼야 한다는 것이다. 정규형에 성능 문제가 발생했을 때만 반정규형을 고려해야 한다.

반정규화 기법으로는 테이블 반정규화(테이블 병합, 테이블 분할, 테이블 추가), 컬럼 반정규화(중복 컬럼 추가, 파생 컬럼 추가, 이력테이블 컬럼 추가, PK에 의한 컬럼 추가, 응용시스템 오작동을 위한 컬럼 추가), 관계 반정규화(관계 추가, 관계 삭제) 등이 있다.

3. 데이터베이스 성능

우리가 흔히 얘기하는 성능에는 크게 두 가지 종류가 있다. 조회 성능과 입력 성능이다. 빠르게 추출해야 하는 조회 성능과 빠르게 저장해야 하는 입력 성능은 다르므로 막연하게 성능이라고 표현하면 의미를 정확하게 전달하지 못할 것이다. 이 두 가지 성능이 모두 우수하면 좋겠지만 데이터 모델을 구성하는 방식에 따라 두 성능이 Trade-Off 되어 나타나는 경우가 많이 있다.[6][16][18]

입력 성능은 많은 트랜잭션을 최대한 빨리 동시에 입력 처리하는 것을 의미한다. 조회 성능에 비해 흔하게 발생하지 않지만 핵심적인 업무와 연관돼 있으므로 간과하면 심각한 문제를 일으킨다. 단순한 성능 문제는 인덱스가 해결책으로 많이 사용되지만 복잡한 문제는 인덱스가 무용지물이 되고 오히려 입력 성능을 저하시킨다.[5]

INSERT 구문에 대한 튜닝은 한계가 있으며 방법이 다양하지도 않다. 한꺼번에 다량이 발생하므로 경합을 줄여주는 방향으로 성능 이슈를 해결해야 한다. 그래야 동시에 가능한 많은 건수를 저장할 수 있다.

대부분 성능 문제라고 하면 조회 성능을 의미할 것이다. 입력 성능과 비교하면 발생하는 빈도수도 훨씬 많으며, 같은 성능 문제가 반복적으로 일어나는 것이 조회 성능이다. 조회도 크게 두 가지로 나눌 수 있다. 한두 건의 소수 데이터를 조회하는 것과 다량의 데이터를 조회하는 것이다. 소수의 데이터를 조회할 때는 성능 문제가 거의 발생하지 않는다. 만약 복잡한 요건으로 사용자가 답답함을 느낄 정도로 느리게 조회되거나, 하루에 조회하는 횟수가 수십만 건이라면 소량의 조회 성능이라도 조회 시간을 최대한 단축해야 한다. 이럴 때 1초와 0.9초는 의미 있는 차이일 수 있다.

소량의 데이터에 대한 조회 성능은 주로 인덱스로 해결되며 대량의 데이터는 스캔 방법과 조인 방법을 사용해서 해결할 수 있다. 조회 성능 문제를 해결할 수 있는 또 다른 방법은 반정규형을 채택하는 것이다. 하지만 반정규형이 정규형보다 성능이 좋아진다는 보장은 없다. 때에 따라 더 나빠질 수 있으므로 무조건 반정규형을 채택하면 얻는 것보다는 잃는 것이 더욱 많을 것이다.

정규화를 하면 중복 데이터가 최소화되고 인스턴스의 크기는 작아진다. 그러면 한 블록에 저장하는 인스턴스는 많아지게 된다. 또한 한 블록에 많은 인스턴스가 존재하면 한번 메모리에 올라온 블록이 다시 사용될 가능성이 커진다. 바로 적중률이 높아진다.

최소한의 블록을 사용해야 한다는 궁극적인 목표가 있는 조회 성능은, 동일한 데이터를 한 블록에 최대한 많이 가질 수 있도록 하는 정규형으로 말미암아 목표를 달성할 수 있다. 최소의 블록을 최대한 자주 사용하도록 하는 것이 조회 성능을 높이는 것이므로 정규형은 분명 조회 성능에 도움을 주게 된다.

조회 성능일 경우 고려해야 할 중요한 요소 중 하나는 화면 구성이다. 화면 구성에 따라 반정규형을 채택할 수도 있다. 화면에 데이터를 보여주는 방법이 테이블 형식이며, 항목이 오른쪽으로 길게 나열되면 데이터 구조도 그에 맞도록 반정규화하는 것이 좋을 때가 많다.[5]

모델링을 수행하면서 모든 조회 요건을 고려한다는 것은 거의 불가능하다. 따라서 우선 해당 엔티티의 조회 요건이 주로 목록인지 명세인지를 확인한다. 목록 위주라면 많은 건수가 조회될 수 있어 조회 성능 문제를 심도 있게 고려해야 한다. 필요하면 중복 속성 등을 사용해 반정규형을 채택한다. 하지만 명세 위주라면 주로 한 건의 데이터만을 조회하는 것이므로 데이터 무

결성만을 고려한다.

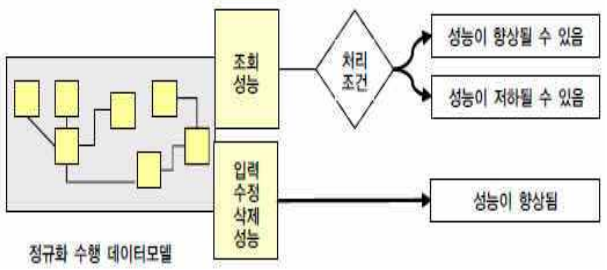
사용자가 원하는 데이터를 정확하고 빠르게 제공하는 것이 데이터 모델링의 목표이므로 성능 문제는 모델링 각 단계에서 검토할 필요가 있다. 또한 성능을 위해 정규화라는 관계형 모델링 원칙을 깨고 반정규형을 사용하는 것이 데이터 무결성을 지키는 것만큼의 가치가 있는지에 대한 검토도 필요하다.

4. 성능 향상 전략

4.1 정규화를 통한 성능 향상

데이터 모델링을 하면서 정규화를 하는 것은 기본적으로 데이터에 대한 중복성을 제거하고 데이터와 관심사별로 처리되는 경우가 많기 때문에 성능이 향상되는 특징을 가지고 있다. 물론 엔티티가 계속 발생하므로 SQL 문장에서 조인이 많이 발생하여 이로 인한 성능 저하가 나타나는 경우도 있지만 이런 부분은 사례별로 유의하여 반정규화를 적용하는 전략이 필요하다.

정규화를 수행한다는 것은 데이터를 결정하는 결정자에 의해 함수적 종속을 가지고 있는 일반 속성을 의존자로 하여 입력, 수정, 삭제 이상을 제거하는 것이다. 데이터의 중복 속성을 제거하고 결정자에 의해 동일한 의미의 일반 속성이 하나의 테이블로 집약되므로 한 테이블의 데이터 용량이 최소화되는 효과가 있다. 따라서 정규화된 테이블은 데이터를 처리할 때 속도가 빨라질 수도 있고 느려질 수도 있는 특성이 있다.



[그림 2] 정규화 수행과 성능의 관계

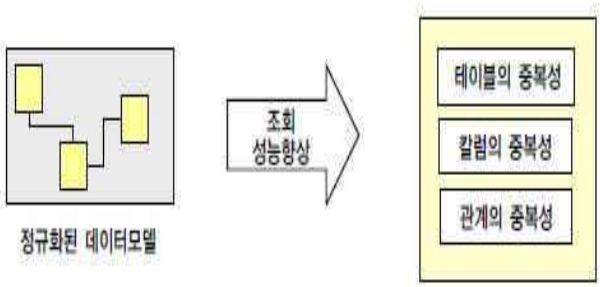
일반적으로 정규화가 잘 되어 있으면 입력, 수정, 삭제의 성능이 향상되고 반정규화를 많이 하면 조회의 성능이 향상된다고 인식될 수 있다. 그러나 반정규화된 테이블에 대해 입력, 수정, 삭제, 조회 성능이 저하되는 일이 빈번하며, 조회 성능을 올리고자 하는 당초의 목적과 달리 데이터의 중복 탓으로 조회 성능이 더 저하되는 경우도 종종 발생한다. 또한 데이터가 중복되니 데이터를 입력하거나 수정할 때 일관성 있게 처리되지 못하여 데이터의 무결성이 깨지기도 한다.

따라서 데이터 모델링을 할 때 반정규화만이 조회 성능을 향상시킨다는 고정관념은 탈피되어야 한다. 정규화가 적용된 데이터 모델이 설계되어야 데이터 처리의 입력, 수정, 삭제, 조회의 성능이 원활하고 데이터 무결성도 보존될 수 있다. 꼭 필요한 반정규화를 제외한 나머지 테이블에 대해서는 정규화의 형태로 유도하여 데이터 처리의 성능과 데이터 무결성을 보장하는 품질이 우수한 모델을 디자인해야 한다.

4.2 반정규화를 통한 성능 향상

반정규화는 정규화된 엔티티, 속성, 관계에 대해 시스템의 성능 향상과 개발과 운영의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링의 기법을 의미한다.

데이터 무결성이 깨질 수 있는 위험을 무릅쓰고 데이터를 중복하여 반정규화를 적용하는 이유는 데이터를 조회할 때 디스크 I/O량이 많아서 성능이 저하되거나 경로가 너무 멀어 조인으로 인한 성능 저하가 예상되거나 컬럼을 계산하여 읽을 때 성능이 저하되기 때문이다.



[그림 3] 중복성의 원리를 활용한 성능 향상

엔티티 타입, 속성, 관계를 반정규화하면 여러 개의 테이블을 읽지 않아도 되므로 SQL 문장이 단순해지고 성능이 향상될 가능성이 많지만, 데이터가 여러 테이블에 걸쳐 존재하므로 똑같은 성격의 데이터 값이 다르게 되는 경우가 발생할 수 있다.

따라서 반정규화를 할 때 가장 중요하게 검토해야 할 기준은 각각의 엔티티 타입과 속성, 관계에 대해 데이터의 정합성과 데이터의 무결성을 우선으로 할지 데이터베이스 구성의 단순화와 성능을 우선으로 할지에 달려있다. 반정규화를 적요할 때는 데이터 무결성의 중요함을 인식하고 데이터 무결성이 충분히 유지될 수 있도록 프로세스 처리 시의 안정성을 먼저 확인해야 한다.

4.3 데이터 모델 단순화를 통한 성능 향상

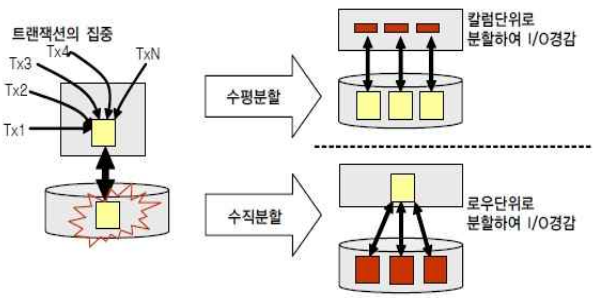
각 모델간의 관계가 연결되어 있지 않고, 통합되어야 할 엔티티 타입이 몇 개로 분리가 되어 있으며, 어떤 엔티티 타입은 과도하게 통합되어 하나의 엔티티 타입으로 표현되어 있으면 이는 전체적으로 품질이 낮은 데이터 모델이 되는 것이다.

이 데이터 모델이 물리적인 테이블로 생성되면 SQL 구문이 길어지고 잦은 조인이 발생할뿐더러 SQL 구문을 한번 실행해서는 명확한 결론이 나오지 않아 두 번 이상 SQL 문장을 실행해야 하는 경우도 생긴다. 결과적으로 낮은 품질로 인한 데이터 모델로 인해 복잡한 SQL 구문이 작성이 되었으며 성능이 저하된 SQL 구문이 작성되게 된 것이다.

따라서 업무 흐름을 정확하게 구별하여 데이터 모델을 생성하고 업무 흐름에 맞게 관계를 연결하여야만 정보 추적성을 보장하고 데이터 무결성도 보장하고, 엔티티 타입이 적절하게 통합되어 SQL 문장에서 테이블 중복으로 인해 발생하는 불필요한 성능 저하 현상을 예방할 수 있다.

4.4 테이블 수평/수직 분할을 통한 성능 향상

아무리 설계가 잘되어 있는 데이터 모델이라고 하더라도 대량의 데이터가 하나의 테이블에 집약되어 하나의 하드웨어 공간에 저장되어 있으면 성능 저하를 피하기가 힘들다. 처리되는 일의 양이 한군데로 몰리는 현상은 전체적인 업무에서 중요한 업무에 해당되는 데이터가 특정 테이블에 있는 경우에 나타난다. 이런 경우 트랜잭션이 분산 처리될 수 있도록 테이블 단위에서 분할의 방법을 적용할 필요가 있다.



[그림 4] 테이블 수평/수직 분할에 의한 성능 향상

많은 컬럼을 가지고 있는 테이블에 대해서는 트랜잭션이 발생할 때 어떤 컬럼에 대해 집중적으로 발생하는지 분석하여 테이블을 쪼개면 디스크 I/O가 감소하게 되어 성능이 개선된다. 즉 트랜잭션을 분석하여 적절하게 1:1 관계로 분리함으로써 성능을 향상시킨다.

테이블에 많은 양의 데이터가 예상될 경우 파티셔닝을 적용하거나 PK에 의해 테이블을 분할할 수 있다. 가장 많이 사용하는 파티셔닝의 기준이 범위 파티션이다. 대상 테이블이 날자 또는 숫자 값으로 분리 가능하고, 각 영역별로 트랜잭션이 분리된다면 범위 파티션을 적용한다. 지점, 사업소, 사업장, 핵심적인 코드 값 등으로 PK가 구성되어 있고 대량의 데이터가 있는 테이블이라면 각각의 값에 의해 파티셔닝이 되는 리스트 파티션을 적용할 수 있다.

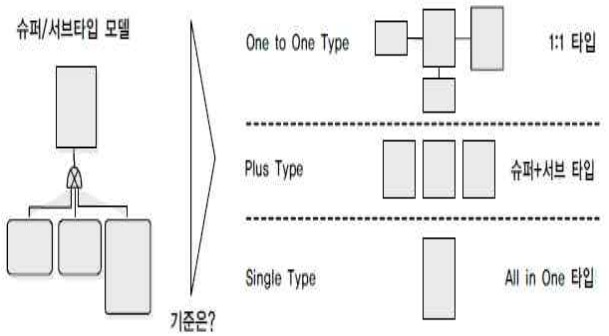
데이터양이 대용량이 되면 파티셔닝의 적용은 필수적이다. 파티셔닝 기준을 나눌 수 있는 조건에 따라 적절한 파티셔닝 방법을 선택하여 성능을 향상시키도록 한다.

4.5 슈퍼타입/서브타입 구분을 통한 성능 향상

슈퍼타입/서브타입 모델은 업무를 구성하는 데이터의 특징을 공통점과 차이점을 고려하여 효과적으로 표현할 수 있기 때문에 자주 쓰이는 모델이다. 이는 공통의 부분을 슈퍼타입으로 모델링하고 공통으로부터 상속받아 다른 엔티티 타입과 차이가 있는 속성에 대해서는 별도의 서브 엔티티 타입으로 구분하여 업무의 모습을 정확하게 표현하면서 물리적인 데이터 모델로 변환을 할 때 선택의 폭을 넓힐 수 있는 장점이 있다.

논리적 데이터 모델에서 설계한 슈퍼타입/서브타입 모델을 물리적인 데이터 모델로 전환할 때 주로 어떤 유형의 트랜잭션이 발생하는지 검증해야 한다. 데이터양이 아주 적은 경우 즉 채 10만 건도 되지 않고 시스템을 운영하는 중에도 증가하지 않는다면 트랜잭션의 성격을 고려하지 않고 전체를 하나의 테이블로 묶어도 된다.

그러나 데이터양이 많고 지속적으로 많은 양이 증가한다면, 슈퍼타입/서브타입에 대해 물리적 데이터 모델로 변환하는 세 가지 유형에 대해 세심하게 적용해야 한다.



[그림 5] 3가지 물리적 모델로 변환 가능한 슈퍼/서브타입 모델

4.6 이력 모델의 구분과 기능성 컬럼을 통한 성능 향상

시간에 따라 구분되는 이력은 발생 이력, 변경 이력, 진행 이력의 형식으로 구분할 수 있다.

발생 이력과 변경 이력은 어떤 특정 시점에 정보가 발생되고, 마지막에 생성된 정보가 빈번하게 이용된다. 진행 이력의 경우 어떤 특정 시점에 정보가 발생되고, 발생된 정보는 계속 그 다음에 새로운 정보가 생성될 때까지 연속적으로 영향을 미치고 있는 형식의 이력 데이터이다.

발생 이력과 변경 이력의 경우는 최종 생성된 데이터를 구분하기 위한 기능성 컬럼이 필요하고, 진행 이력의 경우는 연속적인 특징이 있으므로 생성된 시점과 완료된 시점에 대한 기능성 컬럼이 필요하다는 특징이 있다.

4.7 PK 순서 조정을 통한 성능 향상

PK가 여러 개의 속성으로 구성된 복합식별자일 때 PK순서에 대해 별로 고려하지 않고 데이터 모델링을 한 경우에 성능 저하 현상이 많이 발생한다.

특히 물리적인 데이터 모델링 단계에서는 스스로 생성된 PK 순서 이외에 다른 엔티티 타입으로부터 상속받아 발생되는 PK 순서까지 항상 주의하여 표시하도록 해야 한다.

PK 순서를 결정하는 기준은 인덱스 정렬 구조를 이해한 상태에서 인덱스를 효율적으로 이용할 수 있도록 PK 순서를 지정해야 한다. 즉 인덱스의 특징은 여러 개의 속성이 하나의 인덱스로 구성되어 있을 때 앞쪽에 위치한 속성의 값이 비교자로 있어야 인덱스가 좋은 효율을 나타낼 수 있다. 앞쪽에 위치한 속성 값이 가급적 '=' 또는 최소한 범위 'BETWEEN' '< >'가 들어와야 인덱스를 이용할 수 있다.

4.8 FK 인덱스 생성을 통한 성능 향상

FK를 물리적인 테이블에 걸었을 때는 반드시 FK 인덱스를 생성해야 한다. 물리적인 테이블에 FK를 사용하지 않아도 데이터 모델 관계에 의해 상속받은 FK 속성들은 SQL WHERE 절에서 조인으로 이용되는 경우가 많으므로 FK 인덱스를 생성해야 성능이 좋은 경우가 많다.

FK 인덱스를 적절하게 설계하여 구축하지 않았을 경우 개발 초기에는 데이터양이 얼마 되지 않아 성능 저하가 나타나지 않다가 시스템을 오픈하고 데이터양이 누적될수록 데이터베이스 서버에 심각한 장애 현상을 초래할 수 있다.

그러므로 물리적인 테이블에 FK 제약을 걸었을 때는 반드시 FK 인덱스를 생성하도록 하고, FK제약이 걸리지 않았을 경우에는 FK 인덱스를 생성하는 것을 기본 정책으로 하되 발생되는 트랜잭션에 의해 거의 활용되지 않았을 때에만 FK 인덱스를 지워야 한다.

4.9 CHAR 형식에서 개발 오류 제거를 통한 성능 향상

인덱스 대상 컬럼이 CHAR 형식인 경우 SQL WHERE 절에서 인덱스를 이용하지 못하는 형식으로 컬럼이 비교되는 경우가 많아 성능이 저하된다.

이런 현상을 피하기 위해 개발자는 CHAR 형식으로 지정된 컬럼에 공란을 없애는 함수를 사용한다. 그러면 '인덱스가 걸려 있는 컬럼에 변형이 발생되면 인덱스를 이용할 수 없다'는 전제에 의해 인덱스를 사용할 수 없게 되고, 결과적으로 풀 테이블 스캔이 발생하여 성능이 저하된다.

이를 가변적인 데이터 타입인 VARCHAR 형식으로 데이터 타입을 수정하면 비록 VARCHAR(10)으로 설정되어 있어도 6바이트의 데이터가 들어오면 6바이트만을 점유한다. 그러므로 SQL WHERE 절에서 인덱스 변형이 일어나지 않아 정상적으로 인덱스를 이용할 수 있어 성능이 저하되지 않는다.

CHAR 형을 사용하다가 인덱스를 사용하지 못하면 몇 십초 몇 백초의 성능 저하 현상이 빈번하게 나타난다. 따라서 확실하게 설정한 데이터 타입의 길이를 채울 수 있는 가능성이 100%면 문자형을 사용해도 되지만, 만약 조금이라도 가변적일 수 있다면 가변 형식인 VARCHAR를 사용하는 편이 훨씬 안정적이다.

4.10 효율적인 채번 방법을 통한 성능 향상

대체 식별자인 일련번호 체계를 사용하는 데이터 모델에서는 반드시 채번(PK 값을 증가하는 번호 형식으로 생성하는 것)을 해야 하는데, 이때 채번을 하기 위해 사용된 채번 테이블로 인해 성능 저하가 나타나는 경우가 많다.

이는 채번 테이블을 이용할 때 필연적으로 채번의 대상이 되는 테이블을 수정해야 하고 Commit을 날리지 않은 상태에서 즉, 잠근 현상이 지속되고 있는 상태에서 일련번호를 가지고 있는 다른 테이블에 데이터를

입력해야 한다는 특징 때문이다.

채번은 크게 채번 테이블을 이용하여 일련번호를 증가시키는 방법, 해당 테이블에서 일련번호의 최대 값+1을 바로 가져오면서 입력하는 방법, DBMS에서 제공하는 일련번호 증가 오브젝트를 이용하여 처리하는 방법으로 구분된다.

만약 트랜잭션이 아주 많지 않고 이론적으로 발생하는 데이터 중복 에러에 대해 어플리케이션에서 보완처리할 수 있다면 두 번째 방법이 효과적이다. 트랜잭션 양이 대량으로 발생된다면 세 번째 방법이 효과적이다. 시퀀스 오브젝트의 특징은 읽기만 하면 증가한다. 그래서 가끔 이빨이 빠진 모양처럼 중간이 비는 (1,2,3,5,6,7,8,9…) 경우가 있는데, 이런 데이터 모습이 싫어서 첫 번째 방법을 사용하는 경우가 있다. 그러나 이런 경우는 차라리 테이블의 최대 값을 적용하는 두 번째 방법이 효과적이다.

5. 결 론

정보의 홍수 속에서 범람하는 유효한 데이터를 체계적으로 형상화한다는 것은 쉽지 않다. 이런 상황에서 데이터의 관리는 많은 조직체에서 가장 중요한 작업 중의 하나가 되어 왔고 앞으로도 더 가중될 추세이다. 처리해야 할 정보가 증가함에 따라 데이터의 효율을 최대로 하기 위해 데이터를 어떻게 구성해야 하는가라는 과제는 이미 매우 중요한 문제로 대두되었다.

데이터베이스 응용 프로그램은 원하는 답만 나온다고 해서 역할을 다한 것이 아니며, 적절한 시간 내에 결과를 제공할 수 있어야 하고 다른 프로세스의 작업도 방해하지 않아야 한다. 이를 위해서는 하드웨어, DBMS, 네트워크 등의 모든 영역들이 적절하게 구성되어 있어야 하며, 응용 프로그램 또한 최적화되어 있어야만 한다.

본 연구에서는 VLDB가 된 데이터베이스나 VLDB로 변하고 있는 데이터베이스에서 성능 관리를 하고 성능을 최적화할 수 있는 방법을 DBMS의 관점에서 여러 가지로 알아보았다.

이는 유비쿼터스 시대에 대량으로 발생하게 될 데이터에 대한 관리는 물론 EPCglobal Network를 활용하는 다양한 산업분야에 적용될 수 있으며, 개별 물류센터나 기업에서 물류정보를 효율적으로 관리하는데 활용함은 물론 SCM 전반에 도입하여 시너지 효과를 창출할 수 있을 것으로 기대된다.

6. 참 고 문 헌

[1] 김교중, “대용량 데이터베이스의 성능개선을 위한 튜닝 방법에 관한 연구”, 한남대 정보산업대학원 석사학위논문, 2004.

[2] 김동휘, “대용량 데이터베이스의 튜닝방법 분석 및 적용사례에 관한 연구”, 전남대학교 석사학위논문, 2004.

[3] 김석현, “관계형 데이터베이스를 사용한 효율적 대용량 온톨로지 데이터 처리 방법”, 성균관대학교 석사학위논문, 2011.

[4] 김인식, “대용량 데이터베이스의 효율적인 검색”, 경북전문대학 논문집 제28권, 2009.

[5] 관계형 데이터 모델링, 오픈메이드, 김기창.

[6] 데이터베이스 설계와 구축: 성능까지 고려한 데이터 모델링, 한빛미디어, 이춘식.

[7] 데이터베이스 튜닝, 브레인코리아, 최용락.

[8] 박기현, “효율적인 대용량 데이터베이스 구축을 위한 분해 테이블에 관한 연구”, 동국대학교 석사학위논문, 2004.

[9] 새로 쓴 대용량 데이터베이스 솔루션 I, ㈜엔코아컨설팅, 이화식.

[10] 아는 만큼 보이는 데이터베이스 설계와 구축, 한빛미디어, 이춘식.

[11] 알기 쉽게 해설한 데이터베이스 모델링, 프리렉, 김연홍.

[12] 이상현, 고희수, "SQL서버 대용량 데이터베이스 관리와 모니터링", SQLER 11회 정기 오프라인 기술 나눔 세미나, 2011.

[13] 이종석, 이창호, “시뮬레이션을 이용한 EPCIS의 효율화 방안에 관한 연구”, 대한안전경영과학회지, 제12권 제4호, 2010.

[14] 이창호, 조용철, “EPCIS Event 데이터 크기의 정량적 모델링에 관한 연구”, 대한안전경영과학회지, 제11권 제4호, 2009.

[15] 전준봉, “대용량 데이터베이스 성능개선 방법” 공주대학교 대학원 석사학위논문, 2006.

[16] SQL 전문가 가이드, 한국데이터베이스진흥원.

[17] SQL Server 2000 성능 튜닝, 정보문화사, 하성희.

[18] http://www.dbguide.net

[19] http://www.infomaster.co.kr

[20] http://itmore.tistory.com

저 자 소 개

이 종 석

인하대학교 산업공학과 공학박사 취득. 현재 남서울대학교 산업경영공학과 교수로 재직 중.
관심분야: EPCglobal Network, 시뮬레이션, RFID를 활용한 응용시스템, SCM, DB 등.

주 소 : 충남 천안시 성환읍 매주리 21 남서울대학교 산업경영공학과

이 창 호

인하대학교 산업공학과에서 학사 취득. 한국과학기술원에서 산업공학과 석사, 경영과학과 공학박사 취득. 현재 인하대학교 교수로 재직 중.
관심분야: 물류, RFID, SCM 등.

주 소 : 인천광역시 남구 용현동 253, 인하대학교 산업공학과