1. 반정규화를 통한 성능향상 전략

가. 반정규화의 정의

반정규화(=역정규화) 용어는 조금 다르게 표현되어도 그 의미는 동일하다. 여기에서 반정규화는 '반(Half)'의 의미가 아닌 한자로 반대하다의 의미를 가진 '反'의 의미이다. 영어로는 De-Normalization이다. 비정규화는 아예 정규화를 수행하지 않은 모델을 지칭할 때 사용한다.

반정규화를 정의하면 정규화된 엔터티, 속성, 관계에 대해 시스템의 성능향상과 개발(Development)과 운영(Maintenance)의 단순화를 위해 중복, 통합, 분리 등을 수행하는 데이터 모델링의 기법을 의미한다. 협의의 반정규화는 데이터를 중복하여 성능을 향상시키기 위한 기법이라고 정의할 수 있고 좀 더 넓은 의미의 반정규화는 성능을 향상시키기 위해 정규화된 데이터 모델에서 중복, 통합, 분리 등을 수행하는 모든 과정을 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 성능저하가 예상되거나 칼럼을 계산하여 읽을 때 성능이 저하될 것이 예상되는 경우 반정규화를 수행하게 된다.

중복성의 원리를 활용하여 데이터조회시 성능을 향상시키는 역할을 할 수 있음

[그림 Ⅰ-2-12] 반정규화

기본적으로 정규화는 입력/수정/삭제에 대한 성능을 향상시킬 뿐만 아니라 조회에 대해서도 성능을 향상시키는 역할을 한다. 그러나 정규화만을 수행하면 엔터티의 갯수가 증가하고 관계가 많아져 일부 여러 개의 조인이 걸려야만 데이터를 가져오는 경우가 있다. 이러한 경우 업무적으로 조회에 대한 처리성능이 중요하다고 판단될 때 부분적으로 반정규화를 고려하게 되는 것이다. 또한 정규화의 함수적 종속관계는 위반하지 않지만 데이터의 중복성을 증가시켜야만 데이터조회의 성능을 향상시키는 경우가 있다. 이러한 경우 반정규화를 통해서 성능을 향상시킬 수 있게 되는 것이다.

프로젝트에서는 설계단계에서 반정규화를 적용하게 되는데 반정규화를 기술적으로 수행하지 않는 경우에는 다음과 같은 현상이 발생된다.

성능이 저하된 데이터베이스가 생성될 수 있다.
구축단계나 시험단계에서 반정규화를 적용할 때 수정에 따른 노력비용이 많이 들게 된다.

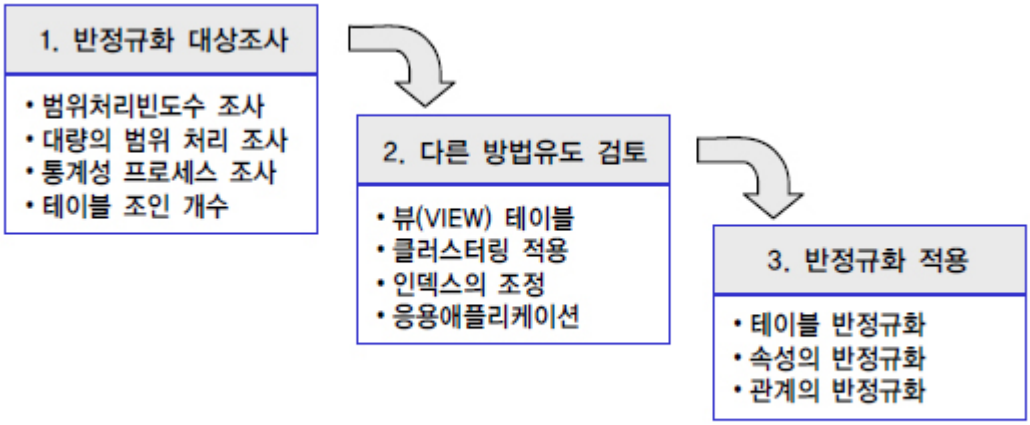
나. 반정규화의 적용방법

반정규화도 하나의 난이도 높은 데이터 모델링의 실무기술이다. 보통 프로젝트에서는 칼럼 중복을 통해서만 반정규화를 수행하게 된다. 칼럼의 반정규화가 많은 이유는 개발을 하다가 SQL문장 작성이 복잡해지고 그에 따라 SQL단위 성능 저하가 예상이 되어 다른 테이블에서 조인하여 가져와야 할 칼럼을 기준이 되는 테이블에 중복하여 SQL문장을 단순하게 처리하도록 하기 위해 요청하는 경우가 많다. 이 때문에 칼럼의 반정규화 유형이 많이 나타나게 된다. 이렇게 무분별하게 칼럼의 반정규화를 많이 하게 되는 것은 데이터에 대한 무결성을 깨뜨리는 결정적인 역할을 하는 경우가 많이 있다.

반정규화에 대한 필요성이 결정이 되면 칼럼의 반정규화 뿐만 아니라 테이블의 반정규화와 관계의 반정규화를 종합적으로 고려하여 적용해야 한다. 또한 반정규화를 막연하게 중복을 유도하는 것만을 수행하기 보다는 성능을 향상시킬 수 있는 다른 방법들을 고려하고 그 이후에 반정규화를 적용하도록 해야 한다.

반정규화를 적용할 때는 기본적으로 데이터 무결성이 깨질 가능성이 많이 있기 때문에 반드시 데이터 무결성을 보장할 수 있는 방법

을 고려한 이후에 반정규화를 적용하도록 해야 한다. 정규화와 반정규화 사이에는 Trade-Off 관계 즉, 마치 저울추가 양쪽에 존재하여 한쪽이 무거워지면 다른 쪽은 위로 올라가는 것처럼 정규화만을 강조하다 보면 성능의 이슈가 발생될 수 있고 반정규화를 과도하게 적용하다 보면 데이터 무결성이 깨질 수 있는 위험이 증가하게 되는 것이다. 따라서 반정규화를 적용할 때에는 데이터 무결성이 중요함을 알고 데이터 무결성이 충분히 유지될 수 있도록 프로세스 처리에 있어서 안정성이 먼저 확인이 되어야 한다.



[그림 Ⅰ-2-13] 반정규화 절차

반정규화를 적용하기 위해서는 [그림 Ⅰ-2-13]에 나타난 것처럼 먼저 반정규화의 대상을 조사하고 다른 방법을 적용할 수 있는지 검토하고 그 이후에 반정규화를 적용하도록 한다.

반정규화의 대상을 조사한다.

일단 전체 데이터의 양을 조사하고 그 데이터가 해당 프로세스를 처리할 때 성능저하가 나타날 수 있는지 검증해야 한다. 데이터가 대량이고 성능이 저하될 것으로 예상이 되면 다음 4가지 경우를 고려하여 반정규화를 고려하게 된다.
- 자주 사용되는 테이블에 접근(Access)하는 프로세스의 수가 많고 항상 일정한 범위만을 조회하는 경우에 반정규화를 검토한다.
- 테이블에 대량의 데이터가 있고 대량의 데이터 범위를 자주 처리하는 경우에 처리범위를 일정하게 줄이지 않으면 성능을 보장할 수 없을 경우에 반정규화를 검토한다.
- 통계성 프로세스에 의해 통계 정보를 필요로 할 때 별도의 통계테이블(반정규화 테이블)을 생성한다.
- 테이블에 지나치게 많은 조인(JOIN)이 걸려 데이터를 조회하는 작업이 기술적으로 어려울 경우 반정규화를 검토한다.

반정규화의 대상에 대해 다른 방법으로 처리할 수 있는지 검토한다.

가급적이면 데이터를 중복하여 데이터 무결성을 깨뜨릴 위험을 제어하기 위하여 반정규화를 결정하기 이전에 성능을 향상시킬 수 있는 다른 방법을 모색하도록 한다.
- 지나치게 많은 조인(JOIN)이 걸려 데이터를 조회하는 작업이 기술적으로 어려울 경우 뷰(VIEW)를 사용하면 이를 해결할 수도 있다. 뷰가 조회의 성능을 향상시키는 역할을 수행하지는 않는다. 다만 개발자별로 SQL문장을 만드는 방법에 따라 성능저하가 나타날 수 있으므로 성능을 고려한 뷰를 생성하여 개발자가 뷰를 통해 접근하게 함으로써 성능저하의 위험을 예방하는 것도 좋은 방법이 된다.
- 대량의 데이터처리나 부분처리에 의해 성능이 저하되는 경우에 클러스터링을 적용하거나 인덱스를 조정함으로써 성능을 향상시킬 수 있다. 클러스터링을 적용하는 방법은 대량의 데이터를 특정 클러스터링 팩트에 의해 저장방식을 다르게 하는 방법이다. 이 방법의 경우 데이터를 입력/수정/삭제하는 경우 성능이 많이 저하되므로 조회중심의 테이블이 아니라면 생성하면 안되는 오브젝트이다. 다만, 조회가 대부분이고 인덱스를 통해 성능향상이 불가능하다면 클러스터링을 고려할 만하다. 또한 인덱스를 통해 성능을 충분히 확보할 수 있다면 인덱스를 조정하여 반정규화를 회피하도록 한다.
- 대량의 데이터는 Primary Key의 성격에 따라 부분적인 테이블로 분리할 수 있다. 즉 파티셔닝 기법(Partitioning)이 적용되어 성능저하를 방지할 수 있다. 인위적인 테이블을 통합/분리하지 않고 물리적인 저장기법에 따라 성능을 향상시킬 수 있는 파티셔닝을 고려해 볼 수 있다. 이 경우는 데이터가 특정 기준(파티셔닝 키)에 의해 다르게 저장되고 파티셔닝 키에 따른 조회가 될 때 성능이 좋아지는 특성이 있다. 따라서 특정 기준에 의해 물리적인 저장공간이 구분될 수 있고 트랜잭션이 들어올 때 일정한 기준에 의해 들어온다면 파티셔닝 테이블을 적용하여 조회의 성능을 향상시키는 것도 좋은 방법이 될 수 있다.
- 응용 애플리케이션에서 로직을 구사하는 방법을 변경함으로써 성능을 향상시킬 수 있다. 응용 메모리 영역에 데이터를 처리하기 위한 값을 캐쉬한다든지 중간 클래스 영역에 데이터를 캐쉬하여 공유하게 하여 성능을 향상 시키는 것도 성능을 향상시키는 방법이 될 수 있다.

반정규화를 적용한다.

반정규화를 적용하기 이전에 사전에 충분히 성능에 대한 고려가 이루어져서 반정규화를 적용해야겠다는 판단이 들었다면 이 때 반정규화의 세 가지 규칙을 고려하여 반정규화를 적용하도록 한다. 반정규화를 하는 대상으로는 테이블, 속성, 관계에 대해 적용할 수 있으며 꼭 테이블과 속성, 관계에 대해 중복으로 가져가는 방법만이 반정규화가 아니고 테이블, 속성, 관계를 추가할 수도 있고 분할할 수도 있으며 제거할 수도 있다.
성능을 향상시킬 수 있는 포괄적인 방법을 적용하여 반정규화를 적용하는 것이 전문화된 반정규화의 기법임을 기억할 필요가 있다.

2. 반정규화의 기법

넓은 의미에서 반정규화를 고려할 때 성능을 향상시키기 위한 반정규화는 여러 가지가 나타날 수 있다.

가. 테이블 반정규화

[표 Ⅰ-2-1] 테이블의 반정규화

기법분류	기법	내용
테이블병합	1:1 관계 테이블병합	1:1 관계를 통합하여 성능향상
	1:M 관계 테이블병합	1:M 관계 통합하여 성능향상
	슈퍼/서브타입 테이블병합	슈퍼/서브 관계를 통합하여 성능향상
테이블분할	수직분할	칼럼단위의 테이블을 디스크 I/O를 분산처리 하기 위해 테이블을 1:1로 분리하여 성능향상(트랜잭션의 처리되는 유형을 파악이 선행되어야 함)
	수평분할	로우 단위로 집중 발생되는 트랜잭션을 분석하여 디스크 I/O및 데이터접근의 효율성을 높여 성능을 향상하기 위해 로우단위로 테이블을 쪼갬(관계가 없음)
테이블추가	중복테이블 추가	다른 업무이거나 서버가 다른 경우 동일한 테이블구조를 중복하여 원격조인을 제거하여 성능을 향상
	통계테이블 추가	SUM, AVG 등을 미리 수행하여 계산해 둠으로써 조회 시 성능을 향상
	이력테이블 추가	이력테이블 중에서 마스터 테이블에 존재하는 레코드를 중복하여 이력테이블에 존재하는 방법은 반정규화의 유형
	부분테이블 추가	하나의 테이블의 전체 칼럼 중 자주 이용하는데 자주 이용하는 집중화된 칼럼들이 있을 때 디스크 I/O를 줄이기 위해 해당 칼럼들을 모아놓은 별도의 반정규화된 테이블을 생성

나. 칼럼 반정규화

[표 Ⅰ-2-2] 칼럼의 반정규화

반정규화 기법	내용
중복칼럼 추가	조인에 의해 처리할 때 성능저하를 예방하기 위해 즉, 조인을 감소시키기 위해 중복된 칼럼을 위치시킴
파생칼럼 추가	트랜잭션이 처리되는 시점에 계산에 의해 발생되는 성능저하를 예방하기 위해 미리 값을 계산하여 칼럼에 보관함. Derived Column이라고 함
이력테이블 칼럼추가	대량의 이력데이터를 처리할 때 불특정 날 조회나 최근 값을 조회할 때 나타날 수 있는 성능저하를 예방하기 위해 이력테이블에 기능성 칼럼(최근값 여부, 시작과 종료일자 등)을 추가함
PK에 의한 칼럼 추가	복합의미를 갖는 PK를 단일 속성으로 구성하였을 경우 발생됨. 단일 PK안에서 특정 값을 별도로 조회하는 경우 성능저하가 발생될 수 있음. 이 때 이미 PK안에 데이터가 존재하지만 성능향상을 위해 일반속성으로 포함하는 방법이 PK의한 칼럼추가 반정규화임
응용시스템 오작동을 위한 칼럼 추가	업무적으로는 의미가 없지만 사용자가 데이터처리를 하다가 잘못 처리하여 원래 값으로 복구하기를 원하는 경우 이전 데이터를 임시적으로 중복하여 보관하는 기법. 칼럼으로 이것을 보관하는 방법은 오작동 처리를 위한 임시적인 기법이지만 이것을 이력데이터 모델로 풀어내면 정상적인 데이터 모델의 기법이 될 수 있음

다. 관계 반정규화

[표 Ⅰ-2-3] 관계의 반정규화

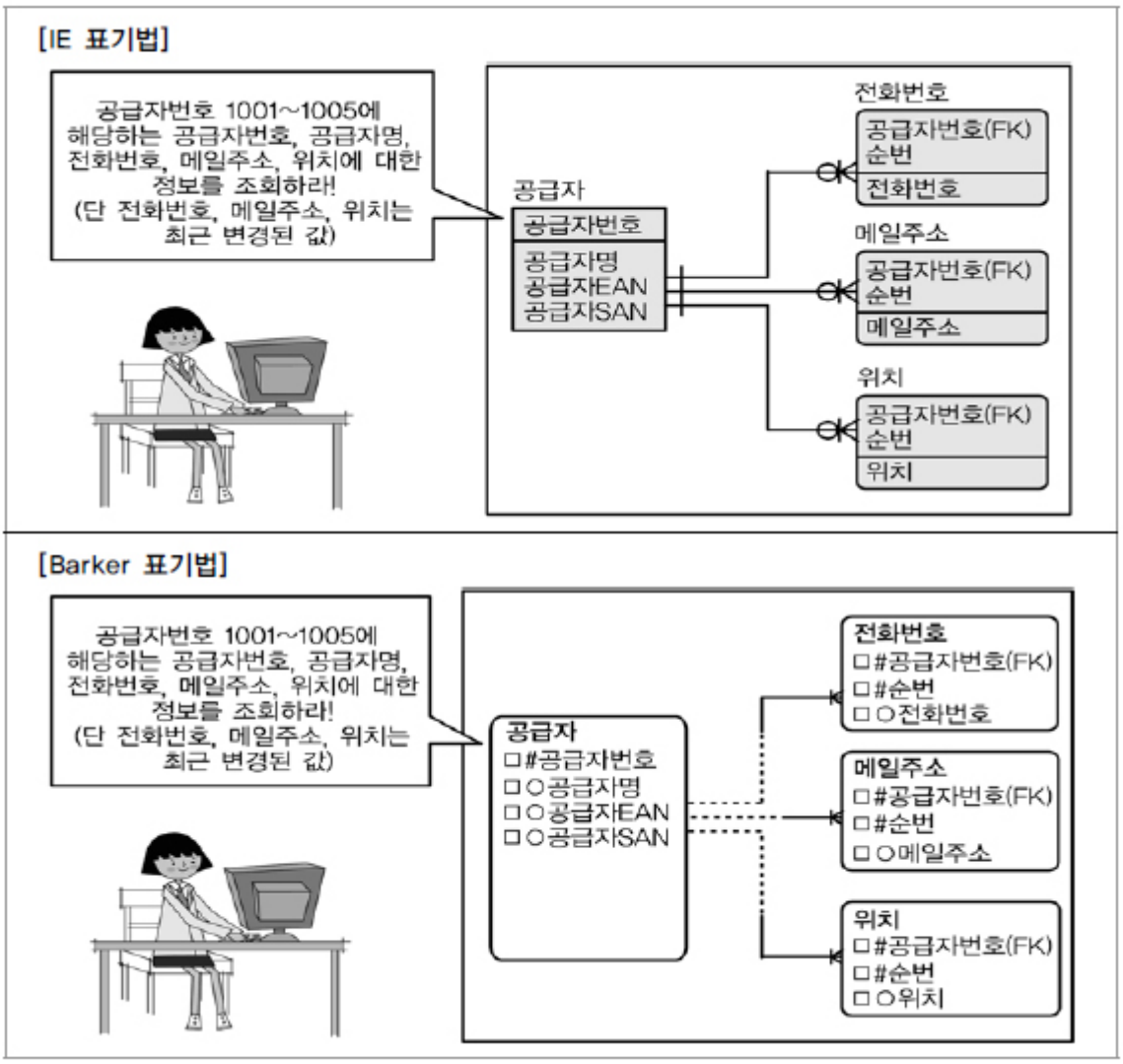
반정규화 기법	내용
중복관계 추가	데이터를 처리하기 위한 여러 경로를 거쳐 조인이 가능하지만 이 때 발생할 수 있는 성능저하를 예방하기 위해 추가적인 관계를 맺는 방법이 관계의 반정규화임

테이블과 칼럼의 반정규화는 데이터 무결성에 영향을 미치게 되나 관계의 반정규화는 데이터 무결성을 깨뜨릴 위험을 갖지 않고서도 데이터처리의 성능을 향상시킬 수 있는 반정규화의 기법이 된다. 데이터 모델 전체가 관계로 연결되어 있고 관계가 서로 먼 친척간에 조인관계가 빈번하게 되어 성능저하가 예상이 된다면 관계의 반정규화를 통해 성능향상을 도모할 필요가 있다

테이블과 칼럼의 반정규화는 데이터 무결성에 영향을 미치게 되나 관계의 반정규화는 데이터 무결성을 깨뜨릴 위험을 갖지 않고서도 데이터처리의 성능을 향상시킬 수 있는 반정규화의 기법이 된다. 데이터 모델 전체가 관계로 연결되어 있고 관계가 서로 먼 친척간에 조인관계가 빈번하게 되어 성능저하가 예상이 된다면 관계의 반정규화를 통해 성능향상을 도모할 필요가 있다.

3. 정규화가 잘 정의된 데이터 모델에서 성능이 저하될 수 있는 경우

[그림 Ⅰ-2-14]는 공급자라고 하는 엔터티가 마스터이고 전화번호와 메일주소 위치가 각각 변경되는 내용이 이력형태로 관리되는 데이터 모델이다. 이 모델에서 공급자정보를 가져오는 경우를 가정해 보자.



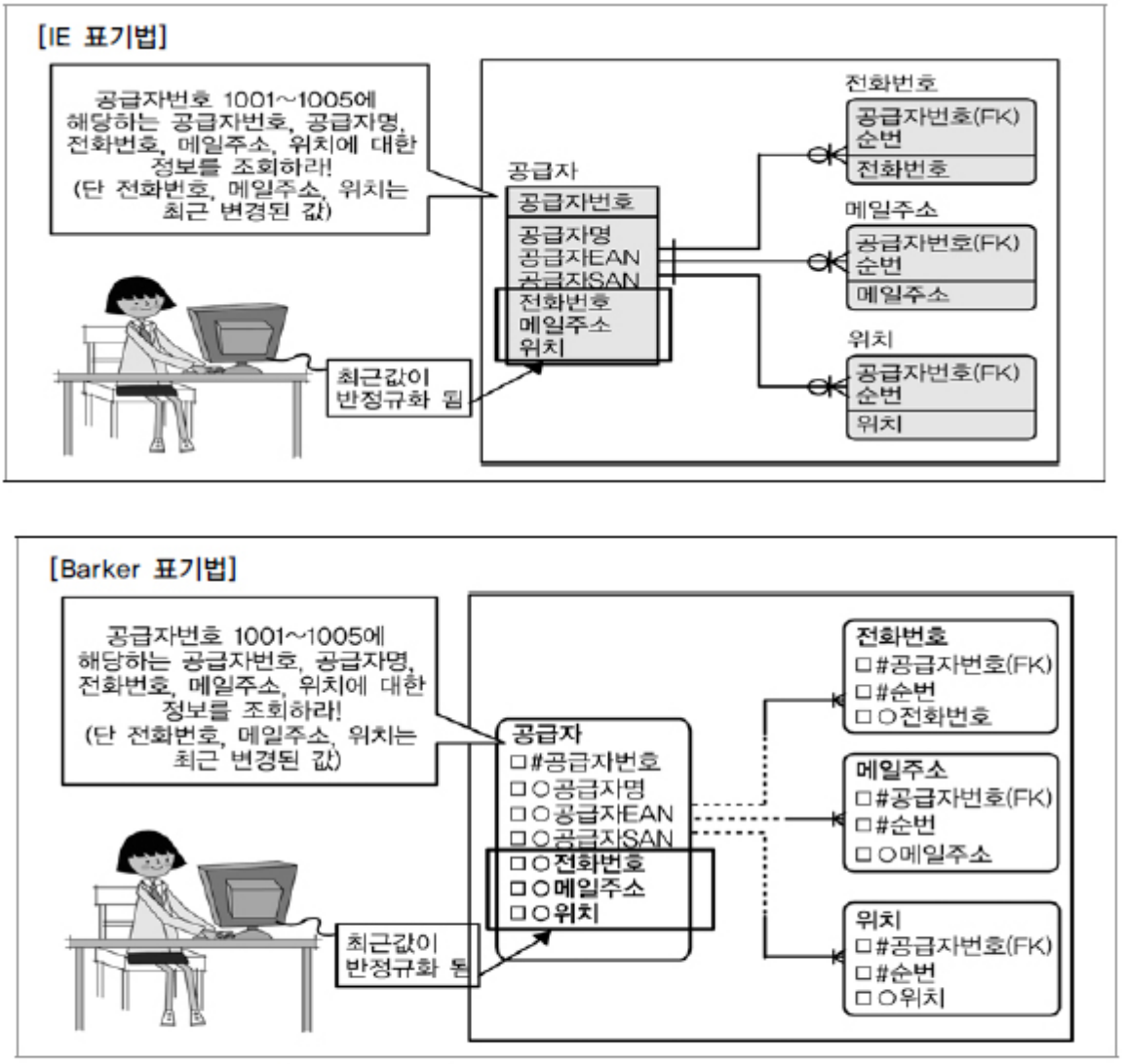
[그림 Ⅰ-2-14] 정규화 모델의 문제점

공급자와 전화번호, 메일주소, 위치는 1:M 관계이므로 한 명의 공급자당 여러 개의 전화번호, 메일주소, 위치가 존재한다. 따라서 가장 최근에 변경된 값을 가져오기 위해서는 조금 복잡한 조인이 발생될 수 밖에 없다.
다음 SQL은 위와 같은 조건을 만족하는 SQL구문이 된다.

SELECT A.공급자명, B.전화번호, C.메일주소, D.위치 FROM 공급자 A, (SELECT X.공급자번호, X.전화번호 FROM 전화번호 X, (SELECT 공급자번호, MAX(순번) 순번 FROM 전화번호 WHERE 공급자번호 BETWEEN '1001' AND '1005' GROUP BY 공급자번호) Y WHERE X.공급자번호 = Y.공급자번호 AND X.순번 = Y.순번) B, (SELECT X.공급자번호, X.메일주소 FROM 메일주소 X, (SELECT 공급자번호, MAX(순

번) 순번 FROM 메일주소 WHERE 공급자번호 BETWEEN '1001' AND '1005' GROUP BY 공급자번호) Y WHERE X.공급자번호 = Y.공급자번호 AND X.순번 = Y.순번) C, (SELECT X.공급자번호, X.위치 FROM 위치 X, (SELECT 공급자번호, MAX(순번) 순번 FROM 위치 WHERE 공급자번호 BETWEEN '1001' AND '1005' GROUP BY 공급자번호) Y WHERE X.공급자번호 = Y.공급자번호 AND X.순번 = Y.순번) D WHERE A.공급자번호 = B.공급자번호 AND A.공급자번호 = C.공급자번호 AND A.공급자번호 = D.공급자번호 AND A.공급자번호 BETWEEN '1001' AND '1005'

정규화 된 모델이 적절하게 반정규화 되지 않으면 위와 같은 복잡한 SQL구문은 쉽게 나올 수 있다. 이른바 A4용지 5장으로 작성된 SQL이 쉽지 않게 발견될 수 있는 것이다.
위의 모델을 적절하게 반정규화를 적용하면 즉, 가장 최근에 변경된 값을 마스터에 위치시키면 다음과 같이 아주 간단한 SQL구문이 작성 된다.



[그림 Ⅰ-2-15] 반정규화를 통한 성능향상 사례

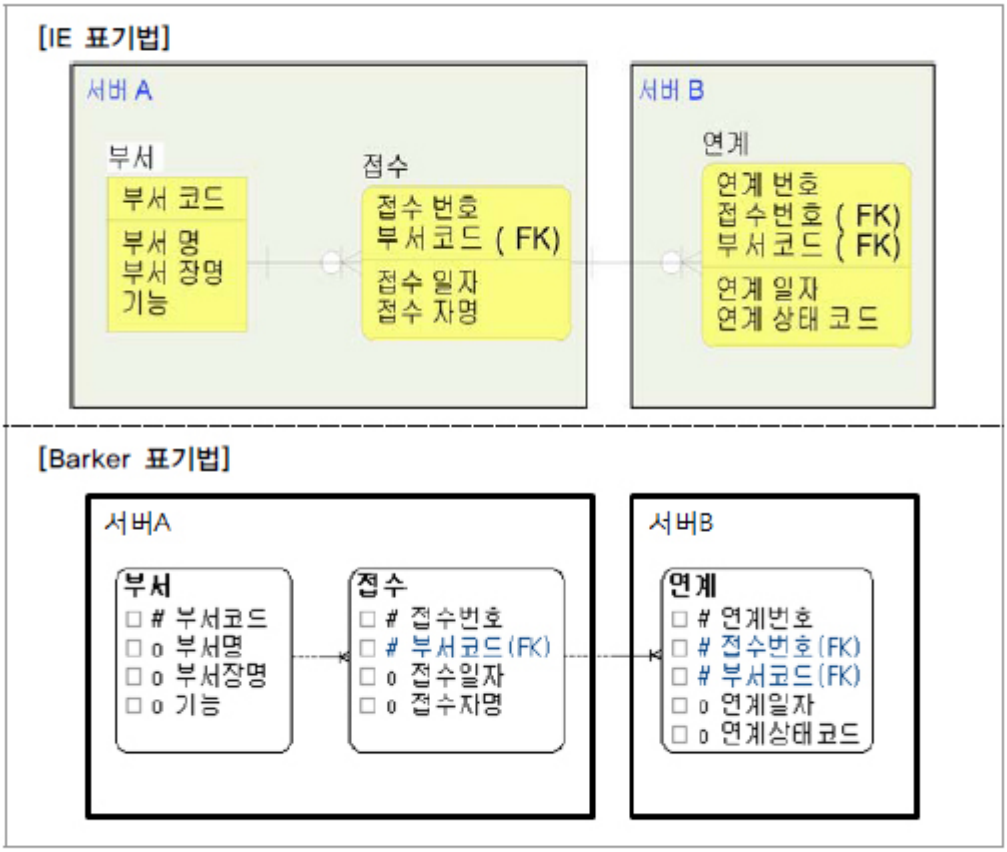
위에서 복잡하게 작성된 SQL문장이 반정규화를 적용하므로 인해 다음과 같이 간단하게 작성이 되어 가독성도 높아지고 성능도 향상되어 나타났다.

SELECT 공급자명, 전화번호, 메일주소, 위치 FROM 공급자 WHERE 공급자번호 BETWEEN '1001' AND '1005'

결과만 보면 너무 당연하고 쉬운 것 같지만 기억해야 할 사실은 위 내용들은 모두 실제로 프로젝트를 할 때도 이와 같이 SQL문장의 성능과 단순성을 고려하지 않고 무모하게 설계되는 경우가 많이 있다는 점이다.

4. 정규화가 잘 정의된 데이터 모델에서 성능이 저하된 경우

업무의 영역이 커지고 다른 업무와 인터페이스가 많아짐에 따라 데이터베이스서버가 여러 대인 경우가 있다. [그림 Ⅰ-2-16]은 데이터베이스서버가 분리 되어 분산데이터베이스가 구성되어 있을 때 반정규화를 통해 성능을 향상시킬 수 있는 경우이다.
서버A에 부서와 접수 테이블이 있고 서버B에 연계라는 테이블이 있는데 서버B에서 데이터를 조회할 때 빈번하게 조회되는 부서번호가 서버A에 존재하기 때문에 연계, 접수, 부서 테이블이 모두 조인이 걸리게 된다. 게다가 분산데이터베이스 환경이기 때문에 다른 서버간에도 조인이 걸리게 되어 성능이 저하되는 것이다.

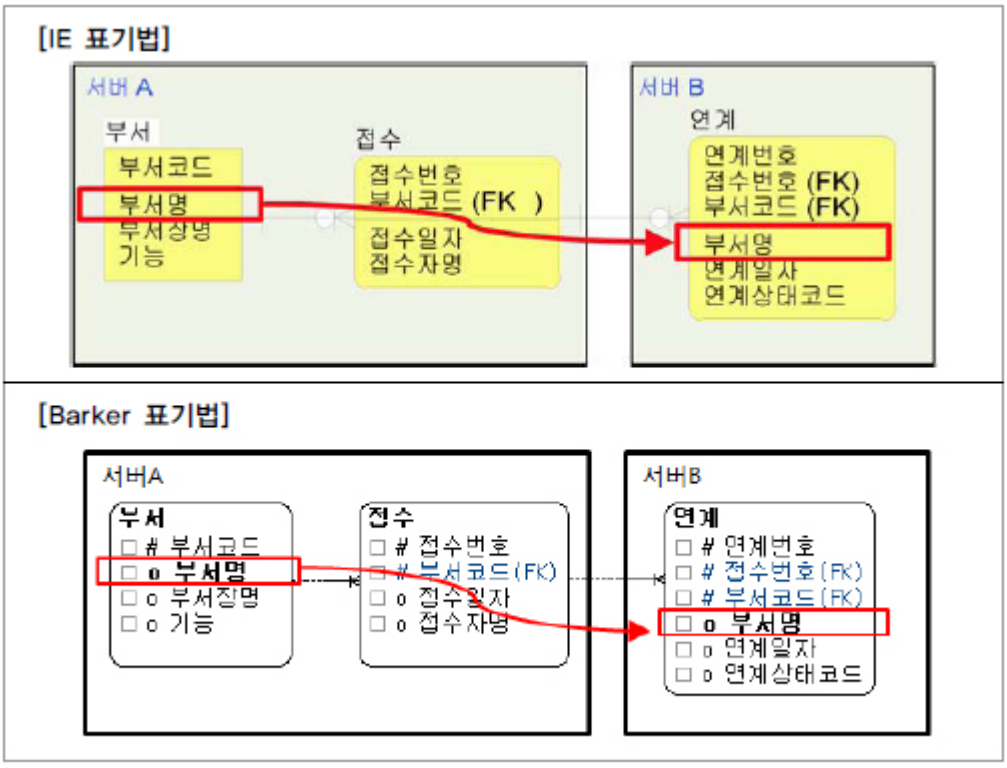


[그림 Ⅰ-2-16] 다른 서버간 정규화된 사례

위의 모델을 통해 서버B의 연계테이블에서 부서명에 따른 연계상태코드를 가져오는 SQL구문은 다음과 같이 작성된다.

SELECT C.부서명, A.연계상태코드 FROM 연계 A, 접수 B, 부서 C <== 서버A와 서버B가 조인이 걸림 WHERE A.부서코드 = B.부서코드 AND A.접수번호 = B.접수번호 AND B.부서코드 = C.부서코드 AND A.연계일자 BETWEEN '20040801' AND '20040901'

Oracle의 경우 DB LINK 조인이 발생하여 일반조인보다 성능이 저하될 것이다.
위의 분산 환경에 따른 데이터 모델을 다음과 같이 서버A에 있는 부서테이블의 부서명을 서버B의 연계테이블에 부서명으로 속성 반정규화를 함으로써 조회 성능을 향상시킬 수 있다.



[그림 Ⅰ-2-17] 다른 서버간 반정규화된 사례

[그림 Ⅰ-2-17]의 모델에 대한 SQL구문은 다음과 같이 작성된다.

SELECT 부서명, 연계상태코드 FROM 연계 WHERE 연계일자 BETWEEN '20040801' AND '20040901'

SQL구문도 간단해지고 분산되어 있는 서버간에도 DB LINK 조인이 발생하지 않아 성능이 개선되었다.
반정규화를 적용할 때 기억해야 할 내용은 데이터를 입력, 수정, 삭제할 때는 성능이 떨어지는 점을 기억해야 하고 데이터의 무결성 유지에 주의를 해야 한다.