정규화는 만병통치약이 아니다

필자는 정규화가 실현할 수 없는 몇가지 것들에 대하여 이야기하는 것을 끝으로 정규화에 관한 연재를 마치고자 한다.(우리가 크게 의존하는 기술이 가진 한계점들을 이해하는 일은 항상 중요하다.) 이러한 본인의 견해가 정규화에 대한 어떠한 종류의 비난도 아니라는 점을 분명히 하고자 한다. 그와 정반대로 필자는 과거에 정규화가 '예술에 가까운 노력에 의해 탄생한' 진정한 과학의 한 부분으로 표현한 적이 있다. 그 노력이라 함은 물론 데이터베이스 설계이다.(실상 우리는 과거에 비해 조금 더 발달된 과학 기술을 접한다. 하지만 데이터베이스 설계는 여전히 주관적인 판단에 따른 문제라는 기본 사실은 아직도 유효하다.) 정규화는 주관적이기보다 객관적이라는 믿을만한 주장이 있지만, 이러한 주장과 전혀 일치하지 않는 데이터베이스 설계상의 몇 가지 경우가 있다는 사실도 여전히 존재한다.

기능 종속의 보존

필자는 우선 기능 종속 보존(FD preservation)의 개념을 소개하고자 한다. 이 개념은 흔히 임의의 관계변수가 다양한 방법으로 비손실 재구성될 수 있지만 그러한 방법들 중 몇몇이 다른 방식들보다 좋을 경우를 말한다.

예를 들어 후보키 EMP#와 기능 종속(FD) EMP#→BUDGET을 지닌 관계변수 EMP {EMP#,DEPT#,BUDGET}를 생각해 보자. 연재 초기에 논의됐던 FD의 전이성(轉移性)으로 인해 FD EMP#→BUDGET 역시 유지된다.(전이(轉移) FD가 점선으로 제시되는 〈그림 1〉을 참조하기 바란다.)

상기의 EMP와 같은 관계변수들은 특정 갱신 이상 현상에 취약점이 있다는 것과 이러한 이상 상태들은 해당 관계변수를 특정 프로젝션들로 비손실 재구성함으로써 회피할 수 있다는 것은 잘 알려져 있는 사실들이다. 우리가 논의하고 있는 예의 경우에 적절한 프로젝션들은 다음과 같다.(둘다 5NF)

ED {EMP#,DEPT#}
CANDIDATE KEY {EMP#}
DB {DEPT#,BUDGET}
CANDIDATE KEY {DEPT#}

필자는 위의 재구성을 "재구성 A"라고 칭하겠다. 이제 EMP의 가능한 5NF 형태의 비손실 재구성이 또하나 존재한다는 사실을 쉽게 알 수 있다.(재구성 B)

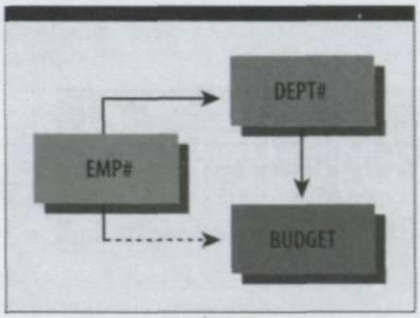
ED {EMP#,DEPT#}
CANDIDATE KEY {EMP#}

EB {EMP#,BUDGET}
CANDIDATE KEY {EMP#}

프로젝션 ED는 A와 B 모두에 있어 동일하다는 점을 주목하자.(참고: 물론 EMP를 이것의 두 프로젝션 {EMP#, BUDGET}와 {DEPT#, BUDGET}로 대체하는 것은 비손실이 아니기 때문에 유효한 재구성이 아니다.)

이제 필자는 몇몇 이유로 인해 재구성 A가 재구성 B보다 더 만족스럽다고 주장한다. 예를 들어, 재구성 B에서는 임의의 부서가 최소한 한명의 고용자가 존재하지 않는 한 예산을 보유하고 있다는 사실을 나타낼 수 없다.

상기의 예를 좀더 자세히 살펴보자. 첫째, 재구성 A의 프로젝션들은 〈그림 1〉에



〈그림 1〉 관계변수 EMP에서의 FDs

서 나타난 실선에 해당한다는 사실을 유념하기 바란다. 그 직접적 결과로, 갱신 작업은 다른 한쪽 프로젝션에 상관없이, 어떤 쪽의 프로젝션으로도 가능하다.(DB로부터 ED로의 참고 제약 조건들이 충족되는 한, 이 후자의 고려 사항은 중요하긴 하지만 현재 우리의 논점과 관계는 없다.)

다시 말해 문제의 갱신 작업이 해당 프로젝션의 내용과 적절할 경우(해당 프로젝션을 위한 후보 키 제약 조건을 위반하지 않아야만 한다는 것만을 의미한다.) 갱신 작업 후의 두 프로젝션들의 결합은 여전히 관계변수 EMP에 있어서 유효할 것이다.(다시 말해 그 결합은 관계변수 EMP상의 FD 제약 조건을 위반할 가능성이 없다는 뜻이다.)

정반대로, 재구성 B에서는 두 프로젝션 중의 하나가 〈그림 1〉의 점선에 해당한다. 그 결과 두프로젝션 어느 한쪽으로의 갱신 작업은, FD DEPT#→BUDGET를 위반하지 않는다는 점을 확인하기 위해서라도 점검되어야만 한다.(만약 같은 부서에 직원이 두 사람이라면 그 둘은 동일한 예산을 보유해야만 한다. 가령 재구성 B 내에서 한명의 직원을 부서 D1에서 D2로 이동시키는 일과 연관된 요소를 생각해 보자.)

재구성 B의 기본적 문제는 FD DEPT#→BUDGET이 두 개의 관계변수와 연계한다는 점이다. 정반대로 재구성 A에서는 전이(轉移) FD EMP#→BUDGET가 관계변수들을 연계한다. 그리고 그 FD는 관계변수들을 연계하지 않는 두 FD EMP#→DEPT#, DEPT#→BUDGET들이 실행될 경우 자동적으로 실행된다. 더불어 이 후자의 두 FD들은 해당 후보 키의 특성 제약 조건들을 실행하는 것 이외에 다른 문제를 수반하지 않기 때문에, 실행하기는 아주 간단하다.

본 논의의 핵심은 관계변수들은 일반적으로 FD를 보존하는 그러한 방식으로 재구성되어져야 한다는 사항이다. 더 상세한 설명으로 다음과 같은 예를 들겠다.

우리가 정규화 하고자 하는 관계변수가 R이라고 가정하자. 그리고 F는 R이 충족시키는 FD들을 위한 원뿔 덮개라고 가정하자. 그런 경우 D라고 명명된 R의 재구성은, F의 모든 FD들이 D의 단하나의 프로젝션과 부합하며 D에는 다른 프로젝션이 존재하지 않는 그러한 형태일 것이다.

참고: 원뿔 덮개(conical cover)의 개념을 간략히 설명하면 FD의 집합 S의 원뿔 덮개는, a) S내의 모든 FD들을 제외한 다른 어떤 것도 포함하지 않고, b) a)에 필요하지 않은 어떠한 FD도 포함하지 않으며, c) 임의의 왼쪽변에 최대 하나의 FD를 포함하는 FD의 집합이다. 가령 상기의 EMP 예의 경우, 이것의 FD들 EMP#→DEPT#, DEPT#→BUDGET, EMP#→BUDGET를 감안할 때, 아래와 같은 FD들은 하나의 변형 불가능한 덮개라는 사실을 쉽게 알 수 있다.(실상, 이것이 유일한 덮개이다.)

EMP# → DEPT#

DEPT# → BUDGET

불행스러운 충돌 문제 하나

S, J, T가 각각 학생, 과목, 교사를 뜻하는 관계변수 SJT {S,J,T}를 생각해보자. 하나의 SJT 열(s,j,t)이 뜻하는 것은 학생 s가 교사 t로부터 과목 j를 수업받는다는 뜻이다. 이 경우 아래와 같이 가정해 보자.

1. 매 과목에 있어 그 과목을 수강하는 학생 각각은 단 한명의 교사로부터 수업을 받는다.

2. 각 교사는 단 하나의 과목만을 가르친다. 그러나 각 과목은 여러명의 교사들에 의해 강의될 수 있다.

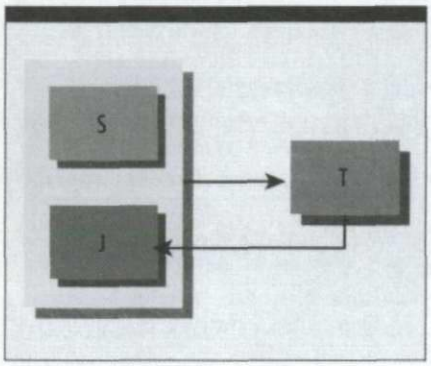
상기의 제약 조건을 따르는 관계변수 SJT의 표본 값은 〈그림 2〉에서 제시된다. 참고: 곧 알게 될 몇몇 이유들로 인해 필자는 이 그림에서 초기 키 칼럼을 이중으로 밑줄치는 본인의 일반적 관행을 벗어난다.

SJT

S	J	T
Smith	Math	Prof. White
Smith	Physics	Prof. Green
Jones	Math	Prof. White
Jones	Physics	Prof. Brown

〈그림 2〉 관계변수 SJT(견본 가치)

어떤 FD들이 SJT내에서 유지되는가? 우리는 조건 1에서 {S,J}→T, 그리고 조건 2로부터 T→J라는 결과를 얻었다. 또한, 각 과목을 여러 교사들이 강의한다는 점은 FD J→T가 유지되지 못한다는 사실



〈그림 3〉 SJT내에서의 FD

을 우리에게 말해준다. 그러므로 그 상황은 〈그림 3〉에서 제시된 바와 같다. 해당 도표의 복잡한 구조를 주목하기 바란다.

SJT는 두 가지 후보 키 {S,J}, {S,T}를 지닌다는 점에 유념하기 바란다. 이 조합들은 모두 SJT의 모든 칼럼들이 기능적으로 이들에 의존한다는 특성을 지니고 있다. 그리고 이러한 특성은 특정 칼럼이 문제의 조합으로부터 배제됐을 경우 더이상 유지되지 않는다. 또한 SJT는 위의 후보 키들 중의 어느 한쪽에 의해 내포되지 않는 추가적 FD T→J를 충족시킨다.

그러므로 SJT는 BCNF의 형태가 아니다. 더불어 모든 non-BCNF 관계변수들과 마찬가지로 SJT는 특정 갱신이상 상태에 대한 취약점을 가지고 있다. 예를 들어 우리가 Jones가 물리학 강의를 수강한다는 정보를 삭제하고자 할 때, Brown 교수가 물리학을 강의한다는 정보를 동시에 잃지 않고서는 그 정보를 삭제할 수가 없다.

항상 그렇듯이, 우리는 이러한 갱신 이상 상태를 프로젝션으로의 비손실 재구성 작업을 행함으로써 회피할 수 있고 위의 경우에 프로젝션들은 다음과 같다.(모두 5NF형태임)

ST {S,T}

CANDIDATE KEY {S,T}

TJ {T,J}

CANDIDATE KEY {T}

이제 우리는 Jones가 물리학을 수강하고 있다는 정보를 프로젝션 ST에 있는 Jones와 Brown 교수에 관계된 열을 없앰으로써 삭제할 수 있다. 그리고 이 경우, 프로젝션 JT에 위치한 Brown 교수와 물리 과목에 관계된 열을 삭제할 필요가 없다.

그러나 문제는 이러한 재구성이 이전에 논의했던 'FD 보존 목적'을 위반한다는 점이다. 좀더 자세히 이야기해서, FD {S,J} -> T는 이제 두가지 관계변수 ST, TJ를 연계하며, 또한 이것은 ST와 TJ에 나타나는 유일한 비정규 FD인 T→J로부터 추론될 수 없다. 그 결과, ST와 TJ는 독립적으로 갱신될 수 없다.

예를 들어, Smith와 Brown 교수에 관계된 열을 ST로 삽입하려는 시도는 틀림없이 거부된다. 왜냐하면, Brown 교수는 물리학을 가르치며 Smith는 이미 Green 교수에게 물리학을 수강하고 있기 때문이다. 그럼에도 불구하고 이러한 사실은 TJ를 조사하지 않고서는 감지되지 않는다.

안타깝게도 이번 논의의 요점은, a) 관계변수들을 궁극적 정규형 또는 BCNF의 형태로 재구성하고, b) FD들을 유지하는 방식으로 관계변수들을 재구성한다는 두가지 목표가 때때로 서로 모순될 경우가 있다는 것이다. 다시 말해 이 두가지 목표를 동시에 달성하는 것이 항상 가능하지는 않다는 것이다.

위의 예에서 다음과 같은 다른 사실들도 생긴다.

* 어떤 관계변수의 설계가 더 좋은 것인지 묻지 말라. 만약 필자가 그것을 안다면 여러분에게 알려드렸을 것이다. 요컨대 단순히 정규화 이론은 어떤 하나의 설계가 보다 효율적이라고 우리에게 알려주는 반면, FD 보존 이론은 다른 설계가 보다 효율적이라고 우리에게 제안한다. 그러므로 그 선택은 몇몇 다른 기준들을 바탕으로 해야만 한다.

* 우리가 two-관계변수 설계를 선택할 경우, 관계변수를 연계하는 FD가 선언되어야만 한다.(시스템이 우리를 위해 이 일을 처리하지 않더라도 문서화의 목적상) 이러한 선언문의 의례적인 형태는 다음과 같을 것이다:

```
FORALL s IN S,t1,t2, IN T,j1,j2 IN J
   IF   EXISTS {S:s,T:t1} IN ST
   AND EXISTS {S:s,T:t2} IN ST
   AND EXISTS {T:t1,J:j1} IN TJ
   AND EXISTS {T:t2,J:j2} IN TJ
   THEN j1  j2
```

여기서 학생, 교사, 과목의 도메인은 각각 S, T, J라 지칭된다고 가정한다. 위의 선언문은 꽤 복잡하다. 그러나 우리가 view를 위해서(혹은 더 일반적으로 어떤 종류의 관계 표현식을 위해서) 후보 키들을 선언할 수 있도록 허용되었다고 가정해보자. 그렇다면 우리는 아마 다음과 같은 선언문을 작성할 것이다:

CREATE VIEW SJT AS (ST JOIN TJ)

CANDIDATE KEY {S,J}

위의 후보 키 선언문은 관계변수 연계

CTXD

C	T	X	D
Physics	Prof. Green	Basic Mechanics	5
Physics	Prof. Green	Principles of Optics	5
Physics	Prof. Brown	Basic Mechanics	6
Physics	Prof. Brown	Principles of Optics	4
Math	Prof. Green	Basic Mechanics	3
Math	Prof. Green	Vector Analysis	3
Math	Prof. Green	Trigonometry	4

〈그림 4〉 관계변수 CTXD의 예

FD를 정의하는 역할을 훌륭히 수행한다. 그러므로 상기의 예는 다른 분야에서 주장했던 필자의 견해, 즉 기초 관계변수의 경우와 마찬가지로 view를 위한 후보 키들의 선언도 가능하다는 결과에 부합하는 또 하나의 논증을 제시한다.

* 여러분이 SJT 예제가 매우 인위적이고, 실제 상황에서 〈그림 3〉에서 제시된 것과 같은 복잡한 종속 구조가 결코 발생하지 않는다고 생각할 경우에 대비해서 다음과 같은 훨씬 낯익은 예를 들겠다:

ADDRESS

{STREET,CITY,STATE,ZIP}

이 관계변수는 FD ZIP -〉 {CITY, STATE}를 충족시킨다. 다시 말해서, 이것은 바로 SJT 예제와 같은 것이다.(S를 STREET로, CITY와 STATE를 J로, ZIP을 T로 생각해 보라.)

중복성 제거

정규화의 광의의 목적은 중복성을 감소시키는 것이다. 그러나 지난달에 제시되었던 관계변수 CTX의 변형인 관계변수 CTXD를 살펴보자. 칼럼 C, T, X, D는 각각 과목, 교사, 교재, 일수(日數)를 의미한다. 열 (c,t,x,d)는 과목 c는 교사 t가 강의할 수 있으며 교재 x를 사용한다는 뜻이다. 그리고 더 나아가, d는 교사 t가 과목 c에서 교재 x를 가지고 소비하는 일수이다. 이제 다음과 같이 가정해 보자.

1. 교사들과 교재들은 상호간에 독립적이다.(지난달 CTX 예제에서와 마찬가지로)

2. 임의의 교사/교재 조합은 과목의 수에 관련 없이 발생할 수 있다.(지난달의 CTX와는 다른 점이다)

〈그림 4〉는 이러한 가설들에 부합하는 표본 CTXD 관계 값을 보여준다.

관계변수 CTXD는 단지 하나의 후보 키로써 조합 {C,T,X}를 지니고 있고, 이것은 단 하나의 비정규 종속 FD {C,T,X} →D를 충족시킨다. 그러므로 이 관계변수는 5NF이다. 그럼에도 불구하고 이것은 상당히 많은 중복성의 문제를 수반한다.

문제는 중복은 프로젝션들을 취한다고 제거되는 것이 아니며, D는 C, T, X의 셋 모두에 의존하므로 이 가운데 어느 하나라도 부족한 관계변수에서는 존재할 수 없다는 것이다. 그러므로 상기의 논의가 주는 메시지는 단지 어느 한 관계변수가 궁극적 정규형(5NF)이라고 해서 갱신 이상 상태나 중복성의 문제가 존재하지 않는 것은 아니라는 사실이다.

참고: 비록 CTXD가 위에서 논의된 FD를 제외한 어떠한 비정규 MVD도 수반하지 않지만, 이것은 C상에 위치한 T와 X의 두 '내장된 MVD'들을 수반한다. 관계변수 R {A,B,...}은 만약 '정규' MVD A↠B가 R의 임의의 프로젝션에서 유지될 경우, A상에 위치한 B의 내장 MVD에 영향을 받는다. 상기의 예에서, 두 내장 MVD들은 CTXD의 명백한 제약 조건들로 지정되어야 한다.

정규화가 모든 문제의 해결책이 아닌 이유

우리의 관심을 순수한 논리적 사항으로 한정한다 하더라도, 정규화는 분명히 만병통치약이 아니다.(물론 필자는 물리적인 사항에 기반한 의견, 즉 성능에 관한 주장은 무시한다.) 그럼 왜 정규화가 모든 문제의 해결책이 아닌가에 대한 이유를 요약하기 위해 다음과 같은 정규화의 목적들에 관해 생각해 보자.

* 중복성의 제거: 그러나 예제 CTXD가 보여주듯이, 정규화는 모든 중복문제를 제거하는데 이용할 수는 없다.

* 갱신 이상 상태의 회피: 정규형은 모든 중복성 문제를 해결하지 못하기 때문에, 갱신 이상 상태 또한 모두 제거할 수 없다.(모든 갱신 이상 상태가 중복성의 문제 때문에 발생하는 것은 아니다.)

* 통합 작업 시행의 단순화: 필자는 이 점에 관해 약간 자세히 말하고자 한다. 특

정 통합 제약 조건들은 다른 제약 조건들을 내포한다는 사실은 일반적인 내용이다. 가령 하나의 평범한 예로서, 제약 조건 SALARY 〉 10,000는 분명히 제약 조건 SALARY 〉 0을 내포한다. 그렇다면 만일 제약 조건 A가 제약 조건 B를 내포한다면, A는 B를 자동으로 실행시킨다는 사실을 알 수 있을 것이다. 그리고 5NF로의 정규화는, 중요하고 통상적으로 발생하는 특정 제약 조건들을 실행하고 또 동시에 그런 제약 조건들을 선언하는 손쉬운 방법이다.

기본적으로 우리가 해야할 일은 후보키의 제약 조건들을 선언하고 실행하는 것뿐이다. 그러면 모든 JD, MVD, FD들은 자동으로 실행될 것이다. 그 이유는 말할 것도 없이 이러한 JD, MVD, FD들이 먼저의 후보 키들에 내포되어 있기 때문이다.

그러나 문제는, a) 모든 제약 조건들이 JD, MVD, 또는 FD가 아니며, b) SJT 예제가 보여주듯이 몇몇 JD나 MVD, 혹은 FD들은 관계변수들을 연계하는(관계변수-spanning) 제약 조건들로 변질될 수 있다.

그러므로 그 경우에 그들은 정규화 공정 기간동안 기술적으로 더 이상 JD나 MVD, 혹은 FD가 아닌 것이다. 다시 말해서 정규화와 FD의 보존이라는 두가지 목적은 때때로 서로 모순될 수 있다.

필자는 아래의 사항도 언급해야 될 줄로 믿는다.

* 개중에는 정규화 자체를 전혀 언급하지 않는 몇몇 데이터베이스 설계 관련 사항들이 있다. 공급자-부품 데이터베이스를 예로 들었을 경우, 우리가 런던의 공급자를 위하여 하나의 관계변수를 가지고 파리의 공급자를 위해 하나의 관계변수를 지니는 대신, 단 하나의 공급자 관계변수를 가져야 한다고 규정하는 것은 무엇인가?

이것은 분명 정규화가 아니다. 적어도 전통적으로 이해되는 프로젝션/결합의 정규화는 아닌 것이다.(다른 연산자들을 기반으로 하는 새로운 종류의 정규화를 개발하는 것이 가능하고, 이론상 새로운 정규화가 방금 공급자와 관련해서 유발되었던 문제들을 해결할 수도 있겠지만 말이다.)

* 여기서 임의의 관계변수에서 똑같이 유효한 몇몇 비손실 재구성들이 존재한다는 점도 지적하고 싶다. 이러한 경우 정규화는 그 중 어느 재구성을 선택할 지에 관해서 별로 언급할 만한 내용을 지니고 있지 않다.(이런 경우 우리는 일반적으로 그냥 경험으로서 가장 적은 수의 프로젝션들을 수반하는 재구성을 선택한다.)

우리는 모두 비정규화가 갱신 작업에 있어 논리적으로 해로운 요소라는 사실을 알고 있다는 점이다. 그러나 이것은 정보검색에 있어서도 해로운 요소이다. 다시 말하면 특정 질문을 공식화하는 작업을 어렵게 한다.

비정규화는 해로울 수 있다

이전에 열거된 많은 문제점에도 불구하고 거의 언제나 완전한 정규화가 아닌 어떤 것도 금기(禁忌)일 수 있다는 믿음을 되새기면서 본 칼럼을 끝맺고자 한다. 완전히 정규화된 설계는 실제 현실의 '훌륭한' 재현, 즉 직관적으로 이해하기 용이하며 미래의 성장을 위한 좋은 기반으로 간주될 수 있다.

여담으로, 훌륭한 top-down 설계 방식론이 완벽하게 정규화 설계를 생성하는 경향이 있다는 이론은 위에 기술한 관계에 있어 아무런 가치가 없다. 여러분은 그 이유가 뭐라고 생각하는가?

마지막으로 한가지 주목할만한 점은 우리는 모두 비정규화가 갱신 작업에 있어 논리적으로 해로운 요소라는 사실을 알고 있다는 점이다. 그러나 이것은 정보검색에 있어서도 해로운 요소이다. 다시 말하면 특정 질문을 공식화하는 작업을 어렵게 한다.

유감스럽게도 오늘 이야기한 '성능 향상을 위한 비정규화'가 존재하는 완벽하지 못한 DBMS들 보다 더한 경우도 때때로 필요하다. 비정규화는 성능 향상에 나쁜 영향을 줄 수도 있다. 실상 '성능 향상을 위한 비정규화'라는 말은 일반적으로 임의의 어플리케이션 하나의 성능 향상을 위해 다른 어플리케이션들의 능률을 저하시킴을 의미한다.

C.J.Date : 필자는 관계형 데이터베이스 시스템의 전문가이며 컨설턴트로서 초청 강사와 자유 기고가로 활동하고 있다.