결합 종속성과 5NF의 최종 정규화 관계

우리는 정규화, 특히 프로젝션으로의 재구성 문제에 관심을 두고 있기에 가능한 프로젝션의 수가 몇인지에 대한 의문에서부터 시작하여야 할 것이다. SPJ 상에서 그 의문의 답은 여덟이며, 세 칼럼 전체당 하나가 존재하는 identity 프로젝션, 두 칼럼당 셋인 바이너리 프로젝션, 각 칼럼당 셋인 unary 프로젝션, 그리고 해당 칼럼이 없이 존재하는 nullary 프로젝션 등이 그것이다.

연관 관계들의 해체

필자는 이전 칼럼에서 제기한 질문에서부터 논의를 시작하고자 한다.(즉, 단지 단 둘뿐이 아닌 세 개, 혹은 그 이상의 프로젝션들을 대치함으로써 정상화될 수 있는 관계변수가 있는가 하는 질문)

참고로 이어지는 논의는 필자의 저서인 데이터베이스 시스템 입문[An Introduction to Database Systems]에서 상당한 양의 자료를 발췌하였다. 또한 필자는 여러분이 특정 관계변수가 어떠한 프로젝션으로 손상됨이 없이 재구성되는 정규화 과정에 대하여 기본적 지식을 갖추고 있다고 가정한다.

먼저, 어떤 공급자(S#)가 어떤 사업(J#)에 어떤 부품(P#)을 공급하는지를 보여주는 관계변수 SPJ{S#, P#, J#}를 생각해 보자. 해당 관계변수는 공급자, 부품, 그리고 사업들간의 복합적인 관계를 나타내고 있으며, 나아가 이것이 모든 실마리(그러므로 이것은 적어도 확실히 Boyce/Codd 정규형 또는 BCNF)임을 주목하기 바란다. 〈그림 1〉은 이러한 관계변수(SPJ 관계)의 예시값을 보여준다.

우리는 정규화, 특히 프로젝션으로의 재구성 문제에 관심을 두고 있기에 가능한 프로젝션의 수가 몇인지에 대한 의문에서부터 시작하여야 할 것이다. SPJ 상에서 그 의문의 답은 여덟이며, 세 칼럼 전체당 하나가 존재하는 identity 프로젝션, 두 칼럼당 셋인 바이너리 프로젝션, 각 칼럼당 셋인 unary 프로젝션, 그리고 해당 칼럼이 없이 존재하는 nullary 프로젝션 등이 그것이다. 참고: 실상, 구성요소 숫자가 n인 집합은 정확히 2의 n승만큼의 가능한 부분집합들을 가지기 때문에 칼럼 수 n의 관계는 정확히 2의 n승만큼의 프로젝션들을 가진다. 물론 identity 프로젝션은 항상 원본 관계와 일치하며, nullary 프로젝션은 항상 TABLE_DUM(원본 관계가 공백일 경우) 내지 TABLE_DEE(그 이외의 경우)이다.

그러나 정규화 작업의 관점에서 볼 때, identity

SPJ

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

〈그림 1〉 Relvar SPJ(관계변수 값)

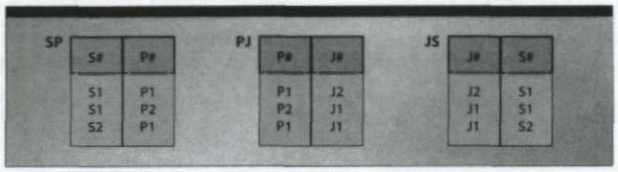
와 nullary 프로젝션들은 분명히 적절치 못하다. 덧붙여, 관계변수의 unary 프로젝션으로의 재구성이 일반적으로 손실적이기 때문에 unary 프로젝션들 또한 일반적으로 적절치 못하다. 그러므로, 상기의 관점에서 볼 때 유일한 관심의 대상은 바이너리 프로젝션일 것이다. 〈그림 1〉의 샘플 데이터에 상응하는 바이너리 프로젝션의 값들은 〈그림 2〉에 제시된다.(그러한 값들을 알기 쉽게 SP, PJ, JS등으로 표기하였다.)

다음으로 〈그림 3〉은 〈그림 2〉에서의 프로젝션 SP와 PJ의 결합(P#의 결합을 통해서)을 보여준다. 표에서 나타난 바와 같이 결합의 결과는 〈그림 1〉의 원본 SPJ 관계의 사본과 하나의 부가적 의사 열(spurious row)로 구성되어 있다. 의사 열의 존재는 〈그림 3〉에서의 결합이 분실된 정보를 포함하고 있다는 것을 의미한다는 사실을 주지해야 한다. 왜냐하면, 도표에 나타난 해당 결합의 결과만으로는 우리는 어느 열이 참 값을 가지고 있고 어느 열이 거짓 값을 가지고 있는지 말할 수 없기 때문이다. 다시 말해서, 그 결합의 결과만으로는 데이터상에 존재하는 업무의 진정한 상태가 실제 상황에서 어떠한지 단언할 수 없다.

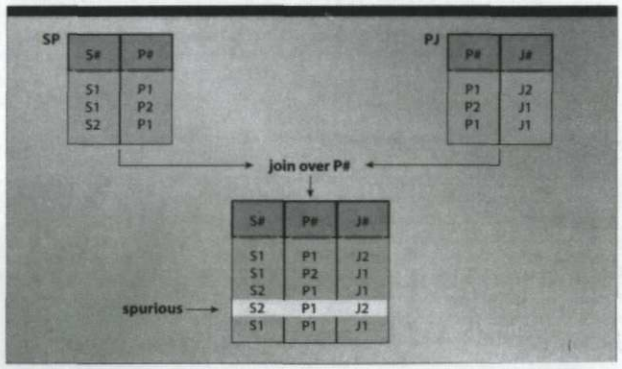
그러나 우리가 〈그림 3〉에서 얻은 결과와 〈그림 2〉에서의 프로젝션 JS(칼럼 J#와 S#의 결합을 통해서)를 결합하면 앞서 말한 의사 열은 사라지며〈그림 4〉 최종 결과는 〈그림 1〉에서의 원본 SPJ 관계와 다시 일치한다. 다시 말해서 상기의 최종 결과는 실제 상황에서의 진정한 사실을 제시하는 것이다.

그러므로 우리는 〈그림 1〉과 SPJ의 관계는 이것의 두 바이너리 프로젝션 SP, PJ와 일치하지 않지만 프로젝션 SP, PJ, JS 전체를 결합한 값과는 일치한다는 사실을 알 수 있다. 더구나, 우리가 첫번째 결합을 위하여 어떤 프로젝션 한 쌍을 선택하더라도 그 최종 결과는 동일하다는 사실을 알 수 있다.

상기의 세가지 경우 각각에서 의사 열은 상이하겠지만 모든 경우에 첫번째 결합은 의사 열을 지니



〈그림 2〉 샘플 데이터에 상응하는 바이너리 프로젝션의 값



〈그림 3〉 프로젝션 SP와 PJ의 결합 값

고 있고 두번째 결합에서 이것은 제거된다. 〈그림 1〉의 관계 SPJ는 비손실적 방식으로 해당 관계에서 도출되는 세가지 프로젝션들로(그러나 그 중 어느 둘만으로는 불가능) 대체될 수 있는 관계의 한 예이다.

관계변수의 재구성

물론 이제까지의 논의는 SPJ 관계변수가 아닌 특정 SPJ 관계에 한정해서 표현되었다 (예를 들면 〈그림 1〉에 나타난 관계 등). 이제 우리의 주의를 〈그림 1〉의 관계가 하나의 가능한 값으로 존재할 수 있는 관계변수 SPJ로 돌려보자. 관계변수가 다음과 같은 통합제약조건에 영향을 받는다고 가정해보자(몇몇 사람들은 이것을 비즈니스 룰이라고 부르기를 원할지도 모른다).

```
FORALL s1,s2 IN S#,p1,p2 IN P#,j1,j2 IN J#
IF    EXISTS {S#:s1,P#:p1,J#:j2} IN SPJ
AND   EXISTS {S#:s2,P#:p1,J#:j1} IN SPJ
AND   EXISTS {S#:s1,P#:p2,J#:j1} IN SPJ
THEN  EXISTS {S#:s1,P#:p1,J#:j1} IN SPJ
```

필자는 여기서 공급자, 부품, 사업 고유번호의 범

주가 S#, P#, J#로 지칭된다고 가정한다. 차후에 명확해질 이유 때문에 필자는 이러한 제약조건을 제약조건 3D라고 부르겠다. 제약조건 3D는 대강 이러한 것이다: 어느 적합한 SPJ 관계 값이 주어졌을 때, 만약 어떤 열들(rows)이 해당 관계 내에 존재한다면 다른 부가적 열 또한 해당 관계 내에 존재해야만 한다. 혹은, 좀더 상세히 이야기해서 s1과 s2가 공급자의 번호일 경우, p1과 p2는 부품의 번호들이며, j1과 j2는 프로젝트의 번호들이다. 그리고 만약 열 (s1,p1,j2), (s2,p1,j1), (s1,p2,j1) 모두가 특정 SPJ 관계 값에 나타난다면, 열 (s1,p1,j1) 또한 해당 SPJ 관계 값에 나타나야만 한다. 다른 방법으로 설명하면

IF EXISTS some SPJ row saying s1 supplies p1
AND EXISTS some SPJ row saying p1 is supplied to j1
AND EXISTS some SPJ row saying j1 is supplied by s1
THEN EXISTS some SPJ row saying s1 supplies p1 to j1

현실에서 이러한 제약 조건 3D는 무엇을 의미할까? 한 예를 이용하여 이 물음에 좀더 확실히 답해본다면 다음과 같다. 제약 조건이 관계변수 SPJ가 대표하는 현실의 부분에서 다음과 같은 사실을 가정한다고 하자. 가령 예를 들어

a. Smith는 멍키 렌치를 공급하고,

b. 멍키 렌치는 Manhattan 프로젝트에서 사용되며,

c. Manhattan 프로젝트는 Smith에 의해 장비를 제공받는다,

d. 그렇다면 Smith는 Manhattan 프로젝트에 멍키 렌치를 공급한다.

일반적으로 상기한 a, b, c에서 얻은 결과들이 d의 내용을 내포하지 않는다는 점을 이해하는 것은 매우 중요하다. 다시 말해 우리는 일반적으로 a, b, c로부터 d를 정확하게 유추할 수 없다. 더 자세히 말해서, 만약 우리가 a, b, c의 내용을 알면, 우리는 Smith가 멍키 렌치를 어떤 프로젝트에 공급하며 (프로젝트 Jz라고 가정하자), 어떤 공급자는(공급자 Sx라고 가정) Manhattan 프로젝트에 멍키 렌치를 공급하며, Smith는 어떤 장비(장비 Py)를 Manhattan 프로젝트에 공급한다는 사실을 추론할 수 있을 것이다. 그러나 우리는 Sx가 Smith인지, Py가 멍키 렌치인지, 또는 Jz가 Manhattan 프로젝트인지 정확히 추론해 낼 수 없다.

참고로 이와 같은 잘못된 추론은 간혹 연결 함정이라고 불리는 것의 예이다. 그러나 우리가 논의한 상기의 경우에는 우리의 추론을 결과적으로 유효하게 하는 제약 조건 3D라는 비즈니스 룰이 있기 때문에 이러한 함정은 존재하지 않는다.

이제 제약 조건 3D가 관계변수 SPJ의 모든 합당한 값이 해당 관계의 세바이너리 프로젝션들의 결합과 항상 일치한다는 사실을 의미하는 것에 주목하자. 그 이유가 궁금하다면 첫째, s1은 p1을 공급한다는 내용을 지닌 SPJ 열이 존재한다는 것은, 보다 자세히 말하자면 상응하는 SP 프로젝션은 s1과 p1을 연계시켜 주는 열(row)을 포함하고 있다는 점을 뜻한다는 사실에 주목하자.

그와 같이, p1은 j1으로 공급된다는 내용을 지닌 SPJ 열이 존재한다는 것은, 보다 자세히 말해서 상응하는 PJ 프로젝션은 p1과 j1을 연계 시켜주는 열을 포함하고 있다는 뜻이며, j1이 s1에 의해 공급된다는 내용을 지닌 SPJ 열이 존재한다는 것은, 더 정확히 이야기해서 상응하는 JS 프로젝션이 j1과 s1을 연계시켜주는 하나의 열을 지니고 있다는 뜻인 것이다. 그러므로 우리는 제약 조건 3D를 바꾸어 말할 수 있는 것이다.

IF EXISTS an SP row linking s1 and p1
AND EXISTS a PJ row linking p1 and j1
AND EXISTS a JS row linking j1 and s1
THEN EXISTS an SPJ row linking s1,p1 and j1

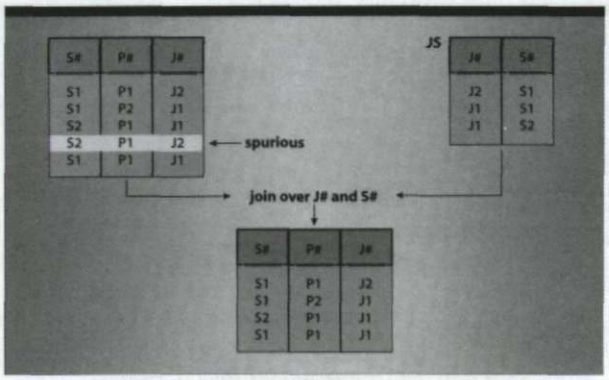
여담으로 필자는 역 포함관계(reverse implication)—즉, SPJ가 열 (s1,p1,j1)을 포함하고 있고, SP, PJ, JS는 열 (s1,p1), (p1,j1), (j1,s1)을 각각 포함하고 있을 경우—는 제약 조건 3D 자체가

유지되는지의 여부에 상관없이 항상 참의 값을 가진다는 점을 지목한다.

마지막으로, 만약 SP, PJ, JS가 각각 열 (s1, p1), (p1, j1), (j1, s1)을 포함하고 있다면 그들의 결합은 열 (s1, p1, j1)을 분명히 포함하고 있을 것이다. 더불어 제약 조건 3D는 해당 열이 상응하는 SPJ 값 내에서 합법적인 열 임을 우리에게 알려준다. 더 나아가, 해당 결합 내에 나타나는 열들은 {S#,P#} 의 부분이 SP 내에 나타나며, {P#, J#} 부분은 PJ 내에 나타나고, {J#, S#} 부분은 JS 내에 나타나는 열들 뿐 임으로, 해당 결합은 비논리적 열들을 포함하고 있지 않다는 사실을 추론 할 수 있다. 달리 말해서, 관계변수 SPJ의 모든 합법적 값들은 제약 조건 3D하에서 해당 관계변수 SPJ의 바이너리 프로젝션 SP, PJ, JS 중의 어떤 둘인 것이다. 또한 그러므로 관계변수 SPJ는 두가지가 아닌 세가지 프로젝션들로 비손실 재구성, 즉 다시 말해 정규화 될 수 있다는 사실이 도출된다.

결합 종속성

필자는 처음에 언급되었던 문제점에 관하여 이제 답을 했다. 그러나 그 외에도 독자에게 드릴 유익한 조언들이 훨씬 더 많이 남아있다. 첫째, 어색하지만 사용하기 편한 용어 3-decomposable을 소개하겠다. 필자는 관계변수 SPJ을 3-decomposable이라 칭하고자 한다. 그것은 관계변수 SPJ가 앞서 말한 세가지 프로젝션으로 비손실 재구성 될 수 있고 그보다 적은 수의 프로젝션으로는 가능하지 않음을 의미한다. 그러므로 필자가 이제까지 제시한 내용은 제약 조건 3D는 관계변수 SPJ가 3-decomposed 될 수 있다는 사실을 내포한다는 것이다. 덧붙여, 그 반대, 즉 관계변수 SPJ가 3-decomposed 될 수 있다는 사실은 제약 조건 3D의 의미를 내포한다는 것 또한 참이라는 사실을 쉽게 알 수 있다. 만약 열 (s1, p1, j1), (s2, p1, j1), (s1, p2, j1)이 SPJ 내에 있을 경우, 열 (s1, p1), (p1, j1), (j1, s1)은 반드시 SP, PJ, JS내에 각각 존재하고, 그러므로 열 (s1, p1, j1)은 그들의 조합 내에 틀림없이



〈그림 4〉 〈그림 3〉의 결과와 〈그림 2〉의 프로젝션 JS와의 결합 값

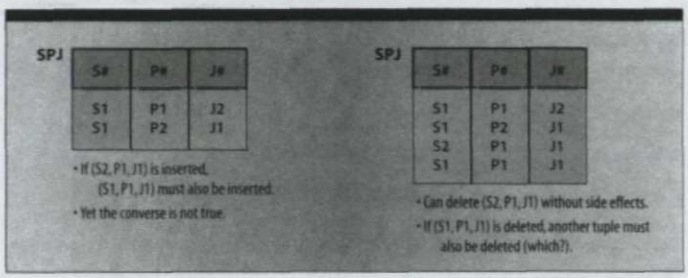
존재하며, 또한 그러므로 열 (s1, p1, j1)은 반드시 SPJ 내에 존재한다(왜냐하면 SPJ는 3-decomposable이며, 따라서 상기의 결합은 SPJ와 일치하기 때문이다).

그러므로 제약 조건 3D는 오직 관계변수 SPJ가 3-decomposable일 경우에만 유지된다. 그리고 상기의 진술은 제약 조건이 오직 관계변수 SPJ가 해당 SPJ의 프로젝션들의 특정 결합과 일치할 경우에만 유지된다. 우리는 제약 조건 3D를 결합 종속(join dependency 또는 JD)이라고 부르며, 그 말의 의미는 다음과 같다.

R을 하나의 관계변수로, 그리고 A, B,.....Z를 R이 지닌 칼럼 집합의 다양한 부분집합들로 가정하자. 그렇다면 R은 R의 모든 합법적 값이 A, B,...Z 상의 R의 프로젝션 결합과 일치할 때 결합 종속 *{A, B,....Z}를 충족시킨다.

가령, 우리가 SPJ의 칼럼 집합 {S#,P#}를 SP라고 칭하기로 하고 그와 같은 방식으로 PJ, JS를 칭하기로 동의했을 때, SPJ는 JD *{SP,PJ,JS}를 충족한다. 참고: 여기서의 주석은 연구 자료나 그 이외의 자료들에서 자연 결합(natural join)임을 의미하기 위해서 사용한 "*"의 보편적 사용을 참고했다.

이제 우리는 관계변수 SPJ는 해당 JD *{SP, PJ, JS}와 함께 세가지 프로젝션들로 비손실 재구성 될 수 있다는 사실을 알았다. 문제는 과연 이것이 꼭 비손실 재구성 돼야만 하는가에 대한 의문인데 그에 대한 답은 거의 분명히 '그렇다' 이다.



〈그림 5〉 SPJ의 갱신 상태

관계변수 SPJ가 3-decomposed 되었을 시에는 제거 가능한 갱신 작업상의 문제들(갱신 이상 상태)에 대해 난점이 있다. 이러한 이상 상태들의 몇 가지 예들이 〈그림 5〉에서 제시된다. 3-decomposition 후 각각의 경우에 어떠한 상황이 발생하는지에 대한 문제는 독자들이 스스로 연구할 문제로 남겨놓겠다.

결론

필자는 다음 칼럼에서 이번 논의를 계속하며, 결합 종속성의 개념이 어떻게 최종 정규형으로 이르게 되는지를 보여줄 것이다. 필자는 몇가지 소감을 마지막으로 칼럼을 끝맺을까 한다.

첫째, 관계변수들이 세 프로젝션들로 비손실 재구성 가능하지만 두 프로젝션으로는 그렇지 않다는 사실이 밝혀졌던 초기에 그것은 꽤 놀라운 일이었다. 과거에는 정규화가 항상 정확히 두 프로젝션으로의 재구성을 수반한다고 가정되었고, 그러한 가정은 4차 정규형(fourth normal form, 4NF)까지 정규화의 연구와 실행을 성공적으로 수행했다.

둘째로, 숫자 3에 어떠한 마술이 없음은 물론이다. 다시 말해, 우리는 몇몇 관계변수들이 세 프로젝션으로 비손실 재구성 될 수 있다는 사실을 깨닫는 동시에 다른 몇몇 관계변수들은 네 프로젝션, 또는 그 보다 많은 프로젝션들로 비손실 재구성되며 세 프로젝션으로는 그렇지 못하다는 사실 등을 이해해야 한다.

실제로, Alfred Aho, Catriel Beeri, Jeffrey Ullman은 단지 3-decomposable 관계변수들만 존재하는 것이 아니라, n〉2인 모든 n-decomposable 관계변수들이 존재할 수 있다는 것을 증명했다. n-decomposable이라 함은 해당 관계변수가 n수만큼의 프로젝션으로 비손실 재구성 될 수 있고 n보다 적은 수로는 그렇지 못함을 의미하는 것이다. 아마도 결합 종속성은 어떤 수의 프로젝션들도 허용한다는 사실을 알 수 있을 것이다.

마지막으로, 제약 조건 3D로 다시 한번 돌아가 보자. 필자는 이제 본 제약 조건의 순환성에 관해 지적하고자 한다. 이러한 성질을 예를 들어 설명하면, 만약 몇 개의 열들이 s1을 p1으로 연계시키고, 다른 몇 개의 열들이 p1을 j1으로 연결시키면, 다른 몇 개의 열들은 j1을 다시 s1으로 연결시키는 것을 가리키며, 이것은 하나의 순환인 것이다. 그리고 어떤 열은 틀림없이 s1, p1, j1을 모두 함께 연결시킬 것이다. 하나의 관계변수가 그러한 n 순환을 수반하는 제약 조건을 충족시킬 때 n-decomposable (n〉2일 경우)일 것이다.

C. J. Date

필자는 관계형 데이터베이스 시스템의 전문가이며 컨설턴트로서 강사와 자유기고가로서 활동하고 있다.