(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) Int. Cl.[6]
G06F 17/30

(11) 공개번호 특1999-028677
(43) 공개일자 1999년04월15일

(21) 출원번호	특1997-710000		
(22) 출원일자	1997년12월31일		
번역문제출일자	1997년12월31일		
(86) 국제출원번호	PCT/US1996/08571	(87) 국제공개번호	WO 1997/03406
(86) 국제출원출원일자	1996년06월03일	(87) 국제공개일자	1997년01월30일
(81) 지정국	AP ARIPO특허 : 케냐 레소토 말라위 수단 스와질랜드 케냐 EA EURASIAN특허 : 아르메니아 아제르바이잔 벨라루스 키르기즈 EP 유럽특허 : 오스트리아 벨기에 스위스 독일 덴마크 스페인 핀랜드 프랑스 영국 그리스 이탈리아 룩셈부르크 모나코 네덜란드 포르투칼 오스트리아 스위스 독일 덴마크 스페인 핀랜드 영국 국내특허 : 아일랜드 알바니아 오스트레일리아 바베이도스 불가리아 브라질 캐나다 중국 체크 에스토니아 그루지야 헝가리 이스라엘 아이슬란드 일본 북한		
(30) 우선권주장	8/499,392 1995년07월07일 미국(US)		
(71) 출원인	월 데이타 인코포레이티드 미합중국 워싱턴 커클랜드 노오스 이스트 122번 웨이 11332		
(72) 발명자	가와이 겐지 미국 워싱턴 98115 시애틀 21 애비뉴 노스이스트 7309		
(74) 대리인	장용식, 정진상		

심사청구 : 있음

(54) 대응 세만틱 객체 모델에서 행해진 변경을 반영하도록 기존 관계형 데이터베이스를 수정하는 방법 및 장치

요약

객체 모델은 컴퓨터 시스템의 관계형 데이터베이스에 저장된 데이터에 대한 항목을 표현하는 하나 이상의 세만틱 객체를 포함한다. 각각의 세만틱 객체는각각의 항목에 저장된 데이터를 정의하는 하나 이상의 성분을 갖는다. 객체 모델은 현재 관계형 데이터베이스 스키마에 매핑된다. 사용자가 객체 모델을 변경시킴에 따라 컴퓨터 시스템은 제안된 관계형 데이터베이스를 생성하고, 현재 관계형 데이터베이스 스키마와 기존 관계형 데이터베이스 스키마간의 차이가 판정된다. 관계형 데이터베이스는 현재 관계형 데이터베이스 스키마와 기존 관계형 데이터베이스 스키마간의 차이에 기초하여 대응 객체 모델에서 행해진 변경을 반영하도록 수정된다.

대표도

도2

명세서

기술분야

발명의 분야

본 발명은 일반적으로 컴퓨터 시스템에 관한 것으로, 더욱 상세히는 관계형 데이터베이스를 이용하여 정보를 저장 및 검색하는 컴퓨터 시스템에 관한 것이다.

배경기술

발명의 배경

언제부턴가 대부분의 컴퓨터 사용자는 일정 종류의 정보를 저장 검색할 필요를 가지게 되었다. 전형적으로, 이것은 여러, 상용가능한 데이터베이스 프로그램중의 하나를 사용하여 달성된다. 이들 프로그램은 사용자로 하여금 데이터베이스에 저장될 정보의 유형을 정의할 수 있게 할 뿐만아니라 데이터를 데이터베이스에 입력할 사용자에게 형식을 제공하고 이전에 저장된 정보를 검색하려는 사람에게 리포트를 제공한다.

가장 인기있는 데이터베이스 유형중의 하나는 관계형 데이터베이스로 일컬어진다. 관계형 데이터베이스에서 데이터베이스는 2차원 테이블에 행으로 저장된다. 각각의 테이블은 저장된 데이터의 유형을 정의하는

일 이상의 컬럼을 갖는다. 종래에는, 초보자 또는 비숙련 사용자는 사용자의 의도를 정확히 반영하도록 데이터를 정렬하고 저장하는 방법으로 관계형 데이터베이스 테이블(데이터베이스스키마로도 참조됨)을 생성하는 것이 곤란하였다.

사용자로 하여금 관계형 데이터베이스스키마를 생성할 수 있게 하는 새로운 접근법은 Washington Seattle 소재 Wall Data Incorporated사에 의해 개발된 SALSA™ 컴퓨터 프로그램이다. 이 프로그램은 사용자로 하여금 데이터베이스에 저장되어야 할 데이터의 모델을 생성할 수 있게 한다. 이 모델은 사용자가 데이터베이스에 저장되어야 할 고유 항목으로 생각할 수 있는 사람,주문, 회사등과 같은 완전한 항목을 표현하는 일 이상의 세만틱 객체로 이루어진다. 각각의 세만틱 객체는 세만틱 객체에 대한 정보를 정의하는 일 이상의 속 성 및 둘 이상의 세만틱 객체간의 객체 링크를 갖는다. 일단 사용자가 세만틱 객체 모델을 완성하면, SALSA™ 컴퓨터 프로그램은 모델을 분석하여 컴퓨터에 저장될 수 있는 대응 관계형 데이터베이스 스키마를 생성한다. SALSA 세만틱 객체 모델링 및 스키마 발생 시스템의 세부사항은 본 명세서에 통합되고, 공동 양도되고 계류중인 1993년 10월 29일 출원된 미합중국 특허 출원 제 08/145,997 호에 개시되어 있다.

대부분의 실세계 데이터베이스 응용에서, 저장될 필요가 있는 데이터의 유형은 시간이 지남에 따라 변경 된다. 관계형 스키마가 생성되었을 때 중요하다고 여겨졌던 정보는 사용자에게 소용없는 것으로 판명된다. 현재, 사용자로 하여금 정보를 제거하고,추가하고 또는 데이터베이스의 정보간의 관계를 변경하기 위해 관계형 데이터베이스 스키마를 용이하게 수정할 수 있게 하는 어떠한 상용가능한 컴퓨터 프로그램도 없다. 따라서, 데이터베이스 스키마를 수정하기 위해, 완전하게 새로운 스키마를 정의하고 이미 데이터베이스에 저장된 데이터로 새로운 데이터베이스 스키마를 채우는 번역 프로그램을 사용하는 것이 때때로 필요하다. 전형적으로, 데이터베이스 스키마는 반드시 데이터베이스 모델링 분야의 전문가에 의해 갱신되어야 한다. 이것은 시간낭비이고 소모적일 뿐만 아니라 단지 데이터를 데이터베이스에 저장하고 데이터베이스 관리 시스템의 내부 작업엔 관심이 없는 사용자에게는 짜증나는 일이다.

현재의 데이터베이스 기술에 따른 문제점에 비추어 볼때, 사용자로 하여금 기존의 관계형 데이터베이스 스키마를 수정할 수 있게 하는 컴퓨터 시스템이 필요하다. 이 시스템은 직관적이어야 하며 사용자로 하여금 이전에는 데이터베이스 모델링 전문가의 도움이 필요하였던 정교한 스키마 수정의 구현을 허용하면서 배우기가 용이하여야 한다.

발명의 상세한 설명

발명의 요약

본 발명은 대응 객체 모델에 행해진 변경을 반영하기 위해 기존의 데이터베이스 스키마를 수정하는 컴퓨터 시스템이다. 이 객체 모델은 각각이 객체를 설명하는 일 이상의 성분을 포함하는 일 이상의 객체를 포함한다. 이 객체 모델은 컴퓨터 시스템의 메모리내에 저장된 관계형 데이터베이스 스키마에 관련된다. 사용자가 이 객체 모델을 수정함에 따라, 컴퓨터 시스템은 제안 스키마를 생성한다. 제안 스키마는 기존의 데이터베이스를 정의하는 현재 스키마에 비교된다. 현재 스키마와 제안 스키마사이에서의 변경은 검출되어 변경 리스트에 결합되고 데이터베이스의 값의 테이블에 반영되어야 한다. 테이블이 수정된 순서는 다른 테이블에 의해 참조된 테이블이 참조하는 테이블 이전에 수정되도록 선택된다.

컴퓨터 시스템은 또한 데이터베이스내의 테이블의 칼럼이 데이터베이스내의 다른 테이블로 이동되어야 할 데이터를 포함하는 지를 판정한다. 데이터를 포함하는 칼럼이 삭제되어야 할 때, 데이터베이스내의 다른 테이블에 추가되어야 할 유사한 칼럼에 대해 탐색이 수행된다. 일단 타를 수신할 테이블이 발견되면, 컴퓨터 시스템은 데이터베이스를 적절한 테이블로 이동시키므로서 데이터베이스를 갱신한다.

도면의 간단한 설명

도 1은 사용자로 하여금 기존의 관계형 데이터베이스 스키마를 수정할 수 있게 하는 본 발명에 따른 컴퓨터 시스템의 블록도.

도 2 내지 8은 관계형 데이터베이스가 대응 객체 모델에 변경을 반영하는 법을 나타낸 다이어그램도.

도 9A 내지 9F는 대응 객체 모델에 행해진 변경을 반영하도록 기존의 관계형 데이터베이스를 갱신하기 위해 두 관계형 데이터베이스를 비교하는 본 발명의 컴퓨터 시스템에 의해 수행된 단계를 도시하는 일련의 흐름도.

도 10A 및 10B는 기존의 관계형 데이터베이스를 수정하기 위해 본 발명의 컴퓨터 시스템에 의해 수행된 단계를 도시하는 흐름도.

도 11A는 데이터베이스에 대한 변경이 데이터 무결성을 보존하는 방식으로 본 발명에 의해 순서화되는 법을 도시하는 다이어그램.

도 11B는 최소 종속 순서로 데이터베이스 테이블을 수정하기 위해 본 발명의 컴퓨터 시스템에 의해 수행된 단계를 도시하는 흐름도.

도 11C는 순환관계로 정의된 관계형 데이터베이스 테이블을 도시하는 다이어그램.

도 12는 대응 객체 모델에 행해진 변경을 표현하기 위해 일 다이어그램으로부터 다른 다이어그램으로 데이터가 이동하는 법을 나타낸 도.

도 12A는 다수-대-다수 값 유형 데이터 이동을 나타내는 도.

도 12B는 다수-대-일 링크 유형 데이터 이동을 나타내는 도.

도 13은 데이터베이스내의 데이터가 일 테이블로부터 다른 테이블로 이동할 때 본 발명에 의해 수행된 단

계를 도시하는 흐름도.

도 14A 내지 14C는 이동 경로를 산출하기 위해 본 발명에 의해 수행된 단계를 도시하는 흐름도.

도 15A 내지 15D는 대응 객체 모델에 행해진 변경을 반영하기 위해 기존의 관계형 데이터베이스를 수정하기 위해 본 발명에 의해 수행된 단계를 도시하는 흐름도.

실시예

본 발명의 상기한 관점 및 다수의 이점은 첨부도면과 연결지어, 다음의 상세한 설명을 참조하므로서 더욱 잘 이해되는 바와 같이 더욱 용이하게 인식될 것이다.

바람직한 실시예의 상세한 설명

사용자로 하여금 관계형 데이터베이스를 용이하게 수정할 수 없게 하는 종래기술의 데이터베이스 관리 시스템의 문제점을 해결하기 위해, 본 발명은 대응 객체 모델에 행해진 변경을 반영하도록 관계형 데이터베이스 스키마를 갱신하도록 프로그램된 컴퓨터 프로그램이다.

도 1을 참조하면, 본 발명의 구현을 위한 컴퓨터 프로그램의 블럭도이다. 이 컴퓨터 프로그램은 일반적으로 디스크 드라이브(74)와 같은 영구 저장 수단 및 내부 메모리(72)를 갖는 CPU(70)를 포함한다. 모니터 또는 디스플레이(76)는 내부 메모리(72) 및 궁극적으로는 디스크 드라이브(74)에 존재하는 관계형 데이터베이스에 저장되어야 할 데이터를 표현하는 객체 모델을 참조할 수 있게 하도록 CPU(70)에 연결된다. 사용자는 마우스(80)와 같은 포인팅 디바이스 및 키보드(78)를 이용하여 객체 모델을 생성한다. 객체 모델이 생성된 후, 컴퓨터 시스템은 CPU(70)로 하여금 모델을 분석하게 하고 메모리(72)에 대응 객체 모델을 생성하게 한다.아래에서 설명되는 바와 같이, 객체 모델에 행해진 변경은 분석되고 메모리에 저장된 데이터베이스 스키마는 행해진 변경을 반영하도록 수정된다.

'997호 특허출원에 완전히 개시된 바와 같이, SALSA 컴퓨터 프로그램은 사용자가 테이블, 속성, 교차 키, 외부 키 및 대용 키와 같은 관계형 데이터베이스 개념을 알 필요없이 관계형 데이터베이스 테이블을 생성하 수 있게 하는 단순한 수단을 제공한다.

SALSA 컴퓨터 프로그램에서, 세만틱 객체는 관계형 데이터베이스에 저장된 데이터에 대한 항목을 표현하는 데 사용된다. 각각의 세만틱 객체는 포함하는 속성에 의해 정의된다. 단순 값 속성은 사람의 이름, 나이, 직업등과 같은 항목의 단순 특성을 나타낸다. 그룹 속성은 세만틱 객체에 표현된 객체의 특성을 집합적으로 정의한다. 주소는 사람이 거주하는 곳을 집합적으로 정의하는 거리, 번지수, 도시 및 주를 포함하므로 그룹 속성의 예이다. 객체 링크 속성은 객체 모델의 세만틱 객체간의 관계를 정의한다.

'997호 특허출원에는, 세만틱 객체가 일 이상의 속성을 포함하는 것으로 정의되어 있다. 이들 속성은 단순 값, 그룹 또는 객체 링크 속성이다. '997호 특허출원과 대조적으로, 세만틱 객체의 속성은 성분으로서 참조되고, 세만틱 객체는 단순히 객체로 참조된다. 그러나 이전에 출원된 발명 및 본 발명의 목적을 위해, 용어 객체는 용어 세만틱 객체와 동의어이고 용어 성분은 용어 속성과 동의어로 한다.

'997호 특허출원에 개시된 세만틱 객체 모델링 시스템이 사용자로 하여금 데이터베이스 스키마를 생성할 수 있게 할 지라도, 사용자가 대응 객체 모델에 행해진 변경을 반영하기 위해 기존 스키마를 수정할 수 있는 어떠한 수단도 제공하지 않는다. 본 발명에서, 사용자가 객체 모델에 행할 수 있는 변경은 4 카테고리로 분할될 수 있다. 이들 카테고리는 모델내의 객체에 대한 변경; 객체내의 성분에 대한 변경; 성분내의 특성에 대한 변경; 모델의 객체간의 관계에 대한 변경이다. 객체 모델에서 발생하는 각각의 변경에 대해, 관계형 데이터베이스의 테이블에 대응 변경이 행해져야 한다.

표 1은 사용자가 행할 수 있는 모델내의 객체에 대한 변경과 모델 변경을 구현하기 위해 데이터베이스 스키마에 행해진 대응 변경을 개시한다.

[표 1]

객체 모델 변경	데이터베이스 스키마 변경
새로운 객체 생성	새로운 테이블 생성
객체 삭제	테이블 해지
객체 이름 변경	테이블 이름 변경

'997호 특허출원에 개시된 바와 같이, 객체 모델의 각각의 객체는 컴퓨터 시스템의 메모리내에 존재하는 대응 관계형 데이터베이스 테이블과 쌍을 이룬다.

각각의 테이블은 전형적으로 그것의 대응 객체와 동일 이름을 갖는다. 사용자가 객체를 모델에 추가한다면, 새로운 관계형 데이터베이스 테이블은 데이터베이스 스키마에 추가되어야만 한다. 마찬가지로 객체가 모델로부터 삭제된다면, 그것의 대응 데이터베이스 테이블은 스키마로부터 삭제된다. 마지막으로 사용자가 객체의 이름을 변경하면, 대응 관계형 데이터베이스 테이블의 이름도 따라서 변경된다.

사용자가 데이터베이스에 가할 수 있는 제 2 유형의 변경은 객체내에서 성분을 수정하는 것이다. 표 2는 사용자가 성분에 가할 수 있는 변경과 관계형 데이터베이스의 테이블에서 행해진 대응 변경을 리스팅한다.

[표 2]

성분 변경	데이터베이스 스키마 변경
단일 값을 갖는, 단순 값 또는 그룹 성분을 추가	새로운 컬럼을 추가
단일 값을 갖는, 단순 값 또는 그룹 성분을 삭제	칼럼을 탈락시킴
다중값을 갖는, 단순 값 또는 그룹 성분을 추가	새로운 테이블을 생성
다중값을 갖는, 단순 값 또는 그룹 성분을 삭제	테이블을 탈락시킴
단일 값을 갖는 객체-링크 성분을 추가	새로운 칼럼을 추가, 외부 키로서 정의
단일 값을 갖는 객체-링크 성분을 삭제	외부 키 칼럼을 탈락시킴
다중값을 갖는 객체-링크 성분을 추가	다수_대_다수 관계이면 새로운 교차테이블을 생성; 또는 외부키로서 새로운 칼럼을 정의
다중값을 갖는 객체-링크 성분을 삭제	교차테이블 또는 외부키 칼럼을 탈락시킴
부모-유형 성분을 추가	새로운 칼럼을 추가, 외부 키로서 정의
부모-유형 성분을 탈락시킴	외부 키를 탈락시킴
서브타입-그룹 성분을 추가/삭제	변경 없음

도 2 내지 8은 객체내의 성분에 가해질 수 있는 변경과 데이터베이스 스키마에 행해진 대응 변경을 다이어그램식으로 도시한다. 이해되는 바와 같이, 객체 모델은 컴퓨터 시스템의 디스플레이상에서 수정되고 데이터베이스 스키마 변경은 컴퓨터 시스템의 내부 메모리에서 행해진다.

도 2는 사용자가 단일값 성분 또는 단순값 및 다중값을 갖는 성분을 객체에 추가할 때 관계형 데이터베이스 테이블에 행해진 변경을 도시한다. 객체(100)는 고용인을 표현한다. 객체는 3개의 단일값 성분:'이름', '주소' 및 '사회_보장_번호'를 포함한다. 데이터베이스에 저장된 고용인 객체의 사례를 고유하게 식별하는 성분은 성분 이름의 좌측에 한 쌍의 아스테리스크로 지시된다.

객체(100)는 '997호 특허 출원에 개시된 바와 같이, 고용인 객체(100)는 관계형 테이블(105)로서 데이터베이스내에서 표현된다. 테이블의 이름은 객체(100)의 이름과 매칭된다. 테이블은 3개의 칼럼: 이름, 주소 및 사회_보장_번호를 갖는다. 고용인 테이블내의 사회_보장_번호 칼럼은 고유한 고용인을 참조하기 위해 테이블의 주 키로서 선택된다. 데이터베이스는 테이블의 주 키를 정의하는 칼럼의 레코드를 유지한다. 또한, 대부분의 데이터베이스는 칼럼이 외부 키인지를 데이터베이스의 다른 테이블에 대해 외부 키인지에 대한 레코드를 유지한다.

사용자가 새로운 단일 값, 단순 값 성분을 객체에 추가할 때, 대응 관계형 테이블(105)는 반드시 갱신되어야 한다. 여기서, 사용자는 고용인 객체에 고용인의 생일을 표현하는 '생일'로 레이블링 된 성분을 추가한다. 성분은 단일 값 성분인 데 이는 고용인은 단지 하나의 생일을 가질 수 있기 때문이다. 데이터베이스에 정보를 저장하기 위해, 칼럼(107)은 고용인 테이블(105)에 추가된다. 칼럼(107)은 추가된 성분과 동일 이름을 갖는다. 칼럼(107)은 생일 성분 즉, 날짜에 의해 정의된 유형의 데이터를 유지하도록 정된다.

알 수 있는 바와 같이, 사용자가 객체로부터 단일 값 성분을 제거한다면, 대응 칼럼은 테이블로부터 제거된다. 사용자가 '사회_보장_번호'로 레이블링 된 성분과 같은, 객체의 사례를 고유하게 식별하는 단일 값 성분을 제거한다면, 테이블의 주키도 제거된다. 이 경우에 대용 키 칼럼이 발생되어 테이블에 추가된다.

객체에 다중값을 갖는 성분을 추가한 결과가 도 3에 도시되어 있다. 학생을 나타내는 객체(110)는 3개의 단순값 성분: '학생_id', '주소' 및 '전화번호'를 포함한다. 학생 객체(110)는 객체내의 3개의 성분의 각각 대응하는 3개의 칼럼을 갖는 대응 관계형 데이터베이스 테이블(112)과 연관된다. 도 3에 도시된 예에서, 사용자는 학생 객체(110)에 ' 전공$_{0.5N}$ '으로 레이블링 된 다중값을 갖는, 단일 값 성분을 추가한다. 이 성분은, 학생이 공표된 임의의 전공을 갖지 않음을 지시하는 제로인 카디널리티와 학생이 하나이상의 전공을 가짐을 지시하는 N인 카디널리티를 갖는 것으로서스크린상에 도시된다.

다중값을 갖는 성분은 관계형 데이터베이스내의 별개의 테이블에 저장된다. 따라서, 다중값을 갖는 성분 ' 전공$_{0.5N}$ '을 학생 객체에 추가하는 것은 테이블(114)이 생성되게 할 것이다. 테이블(114)은 추가되는 성분과 동일 이름을 갖는다. 테이블은 3개의 칼럼: 대응 주키를 유지하는 제 1 칼럼, 학생 전공을 저장하는 제 2 칼럼 및 학생 테이블(112)에 외부 키를 저장한다. 특히, 학생 테이블(112)의 주키(즉, 학생_id로 레이블링 된 저장된 값들)는 테이블(114)에서 외부 키로서 사용되므로써 특정 전공과 연관된 각 학생을 판정하기 위해 테이블(114)을 탐색하고 각 학생의 전공을 판정하기 위해 테이블(114)을 탐색하는 메커니즘을 제공한다.

만일 사용자가 객체로부터 다중값을 갖는, 단일 값 성분을 제거한다면, 대응 테이블(114)은 데이터베이스로부터 삭제된다.

사용자가 단일 값을 갖는, 객체 링크 성분을 객체 모델내의 객체에 추가할 때 관계형 데이터베이스 스키마에서 발생하는 변경이 도 4에 도시되어 있다. 여기서, 객체 모델은 관리자를 표현하는 객체(120)와 비서를 표현하는 객체(122)를 포함한다. 객체 링크 성분(118)을 관리자 객체(120)에 추가하는 것은 관리자와 비서간의 관계를 정의한다. 객체 링크 성분(118)은 제로인 최소 카디널리티를 가지며 관리자가 비서를

가질 필요가 없음을 지시하고, 1 인 카디널리티는 관리자가 비서를 1명 가질 수 있음을 지시한다. 객체 링크 성분(118)을 관리자 객체(120)에 위치시키는 것은 대응 객체 링크(124)가 생성되어 비서 객체(122)에 자동으로 추가되어 지게 한다.

테이블(126)은 관계형 데이터베이스에서의 관리자 객체(120)를 표현한다. 이 테이블은 관리자의 이름, 봉급 및 주소를 저장하기 위한 3개의 칼럼을 포함한다. 테이블(128)은 비서 객체를 표현하는 데 사용된다. 테이블(128)은 적어도 특정 비서의 이름, 주소 및 봉급을 저장하는 4개의 칼럼을 갖는다. 또한, 테이블(128)은 비서가 타이핑할 수 있는 일분당 워드수를 저장하기 위해 'Wpm'으로 레이블링 된 칼럼을 포함한다.

단일 값을 갖는, 객체 링크 성분(118)을 관리자 객체(120)에 추가하는 것은 칼럼(130)이 테이블(126)에 추가되어지게 한다. 칼럼(130)은 '이름'으로 레이블링 된 칼럼과 구별하기 위해 '이름_1'로 레이블링 된다. 칼럼(130)은 관계형 테이블(128)에 대해 외부키를 포함한다. 상기한 바와 같이, 선택된 외부키는 대응 테이블에 대한 주키(즉, 이름)를 참조하도록 정의된다.

테이블(126)에 칼럼(130)을 추가하는 것은 데이터베이스 관리 시스템이 관리자와 특정 비서간에 서로의 연관을 판정하기 위해 테이블을 탐색할 수 있게 한다. 일-대-일 관계에서, 외부 키가 어느 테이블에 추가될 수 있음을 유의해야 한다. 일치를 위해, 본 발명은 대칭인 경우, 사용자가 먼저 객체 링크 성분을 위치시키고, 또는 필요 링크를 가지거나 단일 값을 갖는 그룹 성분에 객체 링크를 갖는 세만틱 객체에 대응하는 테이블에 외부 키를 추가한다.

관리자 객체(120)로부터 객체 링크 성분(118)을 삭제하거나 비서 객체(120)로부터 객체 링크 성분(118)을 삭제하는 것은 외부 키 칼럼(130)이 관계형 테이블(126)로부터 제거되어지게 한다.

도 4에 도시된 객체 모델 수정의 예와 대조적으로, 도 5는 사용자가 다중값을 갖는, 객체 링크 성분을 객체에 추가할 때 관계형 데이터베이스에 행해진 변경을 도시한다. 여기서, 객체 모델은 책을 표현하는 객체와 저자를 표현하는 객체를 포함한다. 책 객체(134)에 다중값을 갖는, 객체 링크 성분(138)을 추가하는 것은 책은 다수의 저자를 가질 수 있고 저자는 여러 권의 책을 저술했음을 지시한다. 다중값을 갖는, 객체 링크 성분(138)이 객체(134)에 추가되었을 때, 대응 다중값을 갖는, 객체 링크 성분(140)은 자동적으로 생성되어 저자 객체(136)에 위치된다.

책 객체를 표현하기 위해, 대응 관계형 테이블(144)은 데이터베이스에 생성된다. 테이블은 책 객체(134) 내의 3개의 성분의 각각에 대응하는 3개의 칼럼을 갖는다. 관계형 테이블(146)은 저자 객체(134)에 대응한다. 저자 객체(134)내의 어느 성분도 특정 저자를 고유하게 식별하지 못하므로, 테이블(146)은 대용 키 칼럼을 포함한다. 대용 키 칼럼은 테이블(146)에 대한 주키이다.

객체(136)내의 다중값을 갖는, 객체 링크 성분(138)을 추가하는 것은 교차 테이블(148)의 생성을 야기한다. 교차 테이블은 테이블(144 및 146)의 주키를 유지하는 두 칼럼을 포함한다. 교차 테이블(148)은 데이터베이스 관리 시스템이 데이터베이스내의 임의의 저자를 판정하고 특정 저자에 의해 저술된 책을 판정한다.

책 객체(134)로부터 다중값을 갖는, 객체 링크 성분(138)을 삭제하는 것은 교차 테이블이 데이터베이스 스키마로부터 삭제되어지게 한다.

도 6은 사용자가 객체 모델의 두 객체간에 부모/서브유형 관계를 생성할 때 데이터베이스가 수정되는 방식을 도시한다. 예를들어, 사용자가 고용인을 포함하는 객체(150)과 관리자를 표현하는 객체(152)를 갖는다면, 관리자 객체(152)내에 부모 유형, 객체 링크 성분(154)를 위치시켜 관리자는 고용인 '임'을 지시할 수 있다. 부모 유형 객체 링크 성분은 성분이름의 하부우측에 첨자 'P'로서 지시된다. 부모 유형, 객체 링크 성분(154)를 관리자 객체(152)내에 위치시키는 것은 고용인 객체(150)에 위치되는 대응 서브유형 객체 링크 성분(156)의 삽입 및 생성을 야기한다.

관계형 데이터베이스내에 고용인 객체(150)를 표현하기 위해, 고용인 객체(150)의 두 성분에 대한 값을 저장하는 두 칼럼을 갖는 관계형 테이블(158)이 생성된다. 관리자 객체를 표현하기 위해, 테이블에 대한 주키로서의 역할을 하는 대용 키 칼럼 뿐만 아니라, 관리자 객체내에 각각의 성분에 대한 값을 저장하는 두 칼럼을 포함한다. 관리자 객체(152)에 부모 유형 객체 링크 성분(154)을 추가하는 것은 테이블(160)에 칼럼(162)을 추가하므로서 데이터베이스에 표현된다. 칼럼(162)은 테이블(158)에 대한 외부 키를 포함한다. 상기한 바와 같이, 선택된 외부 키는 바람직하게 대응 고용인 테이블(158)에 대한 주키이다.

관리자 객체(152)로 부터 부모 유형 객체 링크 성분을 삭제하는 것은 테이블(160)로부터 칼럼(162)이 제거되어지게 한다.

객체내에 성분을 변경하는 이외에, 사용자는 성분 특성에 대해서도 변경할 수 있다. '997호 특허 출원에 개시된 바와 같이, 특성은 데이터베이스의 칼럼에 저장될 수 있는 허용가능 값을 정의한다. 표 3은 객체내의 성분 특성에 행해질 수 있는 변경과 데이터베이스에서 발생한 대응 변경을 도시한다.

[표 3]

성분 변경	대응 데이터베이스 객체 스키마 변경
성분 이름 변경	테이블 또는 칼럼 변경
일로부터 다수로 성분의 최대 카디널리티를 변경	칼럼 탈락, 외부 키 및 대용 키를 갖는 새로운 테이블 생성
다수로부터 일로 성분의 최대 카디널리티를 변경	테이블 및 외부 키 탈락; 새로운 칼럼 추가

일로부터 다수로 객체 링크 성분의 최대 카디널리티를 변경	일_대_일 이면 관련 테이블에 외부키를 생성, 또는 다수_대_다수 이면 교차 테이블 생성
성분의 최소 카디널리티를 변경	NULL 제약조건을 변경
성분의 ID 상태(즉, 고유함)를 변경	UNIQUE 제약조건을 변경
단순값 성분의 값 유형을 변경	데이터 유형을 변경
성분의 디폴트 값을 변경	디폴트 값 변경

표 3으로부터 알 수 있는 바와 같이, 사용자가 성분 특성에 행할 수 있는 대다수의 변경은 관계형 테이블의 칼럼에 저장된 데이터 유형 또는 칼럼의 단순한 재정의를 포함한다.

도 7은 사용자가 다중값을 갖는 성분으로서 단일 값을 갖는, 단순 값 성분을 재정의하는 데이터베이스 변경을 도시한다. 여기서, 객체 모델은 학생을 표현하는 객체(170)를 포함한다. 객체(170)는 '학생_id', '이름','주소' 및 '전공$_{0,1}$' 로 레이블링 된 4개의 성분을 포함한다. 전공 성분의 최소 카디널리티는 제로이고 학생이 전공을 공표하지 않음을 지시한다. 전공 성분의 최대 카디널리티는 1이고 학생이 기껏해야 하나의 전공을 가짐을 지시한다. 데이터베이스내에 학생에 대한 데이터를 저장하기 위해, 관계형 테이블(172)은 컴퓨터 시스템에 생성된다. 테이블은 학생 객체내에 성분의 각각을 표현하는 적어도 4개의 칼럼을 갖는다.

만일 사용자가 전공 성분의 최대 카디널리티를 1부터 N 까지 변경하므로써, 학생이 하나 이상의 전공을 가짐을 지시한다면, 테이블(172)은 재포맷팅된다. 특히, '전공'로 레이블링 된 칼럼은 테이블(172)로부터 제거되고 학생의 전공에 대한 다수의 엔트리를 저장하기 위해 새로운 테이블(176)이 생성된다. 테이블(176)은 일 칼럼이 학생의 전공 예를들어, '영어', '수학'등과 같은 학생의 전공을 저장하고, 두 번째 칼럼은 학생 테이블(172)에 대한 외부 키를 보유한다. 마지막 칼럼은 테이블(176)에 대한 대용 키를 보유한다.

전공 성분에 대한 최대 카디널리티를 N으로부터 1로 변경하는 것은 도 7에 도시된 역 프로세스로서 데이터베이스에 구현된다.

도 8은 사용자가 객체 링크 성분의 최대 카디널리티를 1부터 N 까지 변경할 때 발생하는 데이터베이스 변경을 도시한다. 도시된 객체 모델에서, 객체(180)는 대학의 교수를 나타낸다. 교수 객체는 교수에 대해 저장된 데이터인 '이름', '연령' 및 '소속'으로 레이블링 된 3개의 성분을 갖는다. 또한, 교수를 학생 객체에 링크시키는 '지도 학생'로 레이블링 된 객체 링크 성분이 있고 이로서 교수가 학생의 지도교수임을 표현한다. 객체 링크 성분의 최대 카디널리티가 제로라는 것은 교수가 지도 학생을 가지지 않음을 지시한다. 지도 학생 객체 링크 속성의 최대 카디널리티가 1 이라는 것은 교수가 기껏해야 지도 학생을 1명 가짐을 지시한다.

데이터베이스내에 교수 객체를 표현하기 위해, 교수 객체의 3 성분에 대응하는 3 칼럼을 갖는 관계형 테이블(182)이 생성된다. 또한, 테이블(182)은 대응 학생 테이블(186)로부터 외부 키를 보유하는 칼럼을 포함한다.

지도 학생 객체 링크 속성의 최대 카디널리티가 1로부터 N으로 변경될 때, 교수가 다수의 지도 학생을 가짐을 지시하기 위해, 관계형 테이블(182)의 외부 키 칼럼(184)는 제거된다. 관계형 테이블(182)로부터 외부 키를 보유하는 칼럼(188)은 그후 학생에 대한 데이터를 저장하는 관계형 테이블(188)에 추가된다.

두 객체가 서로에 대해 다중값을 갖는링크를 갖는 경우에 두 객체 테이블에 대해 외부 키를 갖는 교차 테이블이 생성된다.

사용자에 의해 행해질 수 있는 객체모델 변경의 마지막 유형은 테이블 제약조건을 변경하는 것이다. 이것은 객체 모델에 명시적으로 도시되진 않았을 지라도, 사용자는 주키로서 선택된 테이블을 변경할 수 있다. 표 4는사용자가 테이블의 주키를 변경할 때 행해져야하는 대응 변경을 개시한다.

[표 4]

테이블 제약조건	대응 테이블 변경
주 키 변경	주 키 변경 및 대응 외부 키 갱신
주 키 유형(데이터/대용) 변경	주 키 변경 및 대응 외부 키 갱신 및/또는 대용 키 변경

테이블내에서 주키에 대한 변경은 임의의 관련 테이블의 대응 외부 키 뿐만 아니라 관계형 데이터베이스 테이블내에서의 칼럼을 재정의하므로써 행해진다.

영구 저장 매체에 저장되었거나 컴퓨터 시스템내의 데이터베이스 스키마가 갱신되는 방법을 판정하기 위해, 사용자에 의해 정의된 바와 같이, 객체 모델에 대응하는 제안된 관계형 데이터베이스 스키마와 현재 데이터베이스 스키마에 대해 이전에 저장된 정의를 비교하는 것이 필요하다. 두 스키마간의 차이는 판정되고 기존의 관계형 데이터베이스는 이에따라 수정된다.

제안된 및 현재 스키마에 대한 표현을 저장하기위해, 본 발명은 컴퓨터의 영구 저장 수단 및 컴퓨터의 내부 메모리에 저장된 데이터 구조를 생성한다. 이 데이터 구조는 테이블내에서 각각의 칼럼에 대한 정의뿐만 아니라 데이터베이스의 각 테이블에 대한 정의도 포함한다. 제안된 스키마의 데이터 구조는 관계형 데이터베이스 스키마를 생성하기 위해 분석된다.

본 발명의 바람직한 실시예에서, 현재 스키마에 대한 표현을 저장하기위해 사용된 데이터 구조는 다음의

C++ 부류의 사례로 이루어진다. 그러나, 컴퓨터 프로그래밍의 당업자는 기타 데이터 구조가 스키마 정의를 저장하기위해 사용될 수 있었던 것을 인식할 것이다. 또한, 기타 프로그래밍 언어도 사용될 수 있다. 아래에 제공된 부류 정의는 부류에 대한 관련 멤버 변수만을 포함함을 유의해야 한다. 부류의 멤버 기능은 명시적이지 않지만 아래에 설명된 본 발명의 설명으로부터 명백하고 아래의 흐름도에 나타나 있다.

```
부류 Schema
{
tableList;     //객체모델의 각 객체에 대응하는 모든 테이블을 리스팅                    //함
};
```

Schema 부류는 객체모델의 각 객체에 대응하는 테이블 정의 리스트를 포함한다. 다중값을 갖는 성분의 사례를 보유하기위해 생성된 테이블 및 교차 테이블은 리스트에 포함되지 않는다. 테이블의 리스트의 각 엘리먼트는 대응 관계형 데이터베이스 테이블에 대한 완전한 정의를 포함한다.

다음의 TableDef 부류는 데이터베이스 스키마내의 각 테이블의 정의를 저장하기위해 사용된다.

```
부류 TableDef
{
table_ID;          //데이터베이스 테이블에 할당된 고유번호
columnList;        //이 테이블(DataColumn,KeyColumn,GroupColumn,
또는 ForeignKey유형으로 된)의 칼럼 엔트리를
//리스팅함)
*parentTable;      //가능하다면 부모테이블에 대한 포인터
tableList;         //이 테이블에 속하는 자식테이블을 리스팅함
relationList;      //다른 테이블에 대한 관계를 리스팅함
indexList;         //이 테이블에 대해 정의된 인덱스를 리스팅함
primaryKeyType;    //TableKeyType에 대한 사용자 정의
primaryKey;        //주키로서 정의된 인덱스
tableName;         //객체이름과 동일할 수 있는 본래 이름
newTableName;      //데이터베이스내에 저장된 실제 테이블 이름
tableType;         //명시적 테이블 유형(즉, 객체,다중값을 갖는
성분
//또는 교차)
hDBObject;         //객체모델의 대응객체에 대한 취급
};
```

알 수 있는 바와 같이, TableDef 부류는 정보를 테이블의 칼럼에 저장하는 멤버 변수, 적용가능하다면 부모 테이블에 대한 포인터, 이 테이블에 속하는 자식 테이블 뿐만 아니라 테이블간의 관계를 포함한다. 더우기, 테이블에 대한 주키 및 키 유형이 상세되었다.

관계형 테이블내의 각 칼럼에 관한 정보는 부류 DataColumn, KeyColumn, GroupColumn 또는 ForeignKey 사례로서 저장된다. 이들 부류는 모두 아래에서 저의되는 공통 베이스 부류, ColumnDef로 관련된다.

```
부류 ColumnDef     //데이터베이스의 각 칼럼에 정보를 저장하기위한
베이스 정보
{
Col_Id;            //데이터베이스의 각 칼럼에 할당된 고유번호
nullAllowed;       //TRUE-선택적, FALSE-요구됨
idStatus;          //고유성:비-고유,고유 또는 주 키
hDBProp;           //모델에서 대응 객체 성분에 대한 취급
colType;           //DataColumn, KeyColumn, GroupColumn
또는 Foreign
//Key
};
```

이 부류의 어떠한 사례도 실제로 생성되지 않는다. 부류는 단지 다음 유도된 부류의 공통 멤버 변수를 저

장하기위해 사용된다.

부류 DataColumn: 공용 ColumnDef

{

valueType;　　//대응 객체모델에 상세된 데이터 유형

dataType;　　//데이터베이스에 표명된 데이터 유형

logicalLength;　　//적용가능할 때 모델에 상세된 데이터 길이

dataLength　;　　//적용가능할 때 데이터베이스에 상세된

//데이터 길이

scale;　　//적용가능할 때 데이터베이스에 표명된

//정밀 길이

};

DataColumn 부류의 사례는 관계형 데이터베이스 테이블내에 데이터를 보유하는 임의의 칼럼에 정보를 저장한다.

만일 칼럼이 테이블의 외부 키의 멤버로서 정의되었다면, 칼럼에 대한 정보는 다음 부류의 사례로서 저장된다.

부류 KeyColumn: 공용 DataColumn

{

*pRefColumn;　　//부모 외부키에 참조된 테이블의 대응 컬럼

//에 대한 포인터

};

대부분의 데이터베이스 프로그램은 관계형 테이블내의 칼럼 그룹을 인식하지 못하므로, 그룹으로서 논리적으로 구성된 칼럼에 대한 정보는 다음 부류의 사례로서 저장된다.

부류 GroupColumnn : 공용 ColumnDef

{

columnList;　　//그룹내의 칼럼 멤버를 리스팅함

//(Data Column, GroupColumnn 또는

//ForeignKey

};

마지막으로, 테이블내의 임의의 외부 키 관계에 관한 정보는 다음 부류의 사례로서 저장된다. 외부키는 테이블의 외부 키가 단일 칼럼 이상의 것을 포함할 수 있기 때문에 GroupColumnn 부류로부터 유도된다. 그러나 외부키에 대한 columnList의 멤버는 KeyColumnn 사례에 재한된다.

부류 ForeignKey: 공용 GroupColumnn

{

*pReferenceTable;　　//외부 키가 발생된 테이블에 대한

//포인터

};

객체 모델로서 생성되고 초기화된 상기한 바와 같은 부류의 사례는 제안된 데이터베이스 스키마를 정의하기 위해 분석된다.

상기한 바와 같은 데이터 구조에 저장된 데이터베이스 스키마는 사용자에 의해 수정되는 객체 모델에 대응한다. 제안된 스키마는 컴퓨터 메모리에 존재함에 따라 데이터베이스를 설명하는 데이터베이스 스키마와 비교된다. 본 명세서를 위해, 데이터베이스에 대한 이전 정의는 현재 스키마로서 참조된다. 현재 스키마를 설명하는 데이터 구조는 컴퓨터의 영구저장 수단 또는 컴퓨터 메모리에 저장된다. 사례가 객체 모델에 대한 일련의 변경을 완료했을 때, 현재 스키마에 대한 데이터 구조는 영구저장 수단으로부터 호출되어 제안된 스키마를 정의하는 데이터 구조와 비교된다.

이제 도9A 내지 9F를 참조하면, 두 데이터베이스 스키마를 비교 하기위해본 발명의 컴퓨터 시스템에 의해 수행된 단계들이 나타나있다. 상기한 바와 같이, 각각의 스키마는 데이터베이스의 테이블, 각 테이블의 칼럼 및 테이블의 임의의 키(주키 또는 외부 키)를 정의하는 데이터 구조로서 정보된다.

단계(200)에서 시작하여, 컴퓨터 시스템은 제안된 스키마 데이터 구조 및 현재 스키마 데이터 구조를 호출하여 컴퓨터의 내부 메모리에 위치시킨다. 단계(204)에서, 루프를 시작하며 여기서 현재 스키마에 대한 tableList에 리스트된 각 테이블이 제안된 스키마의 대응 tableList의 엔트리에 비교된다.
단계(208)에서, 현재 스키마의 테이블이 자식 테이블의 빈 리스트를 갖는지의 여부를 판정한다. 만일, 그 결과가 아니오 이면, 설명될 테이블 비교 루틴은 자식 테이블의 테이블 리스트에 있는 각 테이블을 위

해 호출된다. 이 재귀동작은 빈 tableList를 갖는 테이블이 발견될 때 까지 발생한다.

단계(214)에서, 스키마 비교 루틴은 현재 스키마의 테이블이 제안된 스키마에서 발견되는 지의 여부를 판정한다. 만일, 그 결과가 아니오 이면, 사용자는 객체 모델로부터 다중값을 갖는 성분 또는 객체를 삭제하여야 하며 단계(218)에서 테이블은 관계형 데이터베이스에서 삭제된다.

만일 테이블이 제안된 스키마에서 발견된다면, 단계(216)에서, 비교 루틴은 특정 테이블에 대해 정의된 columnList의 각 칼럼을 분석한다. 단계(220)에서, 컬럼이 유형 GroupColumn으로 된 것인 지의 여부를 판정한다. 만일 그 결과가 예 이면, 단계(216)에서의 비교 루틴이 그룹을 만드는 칼럼을 추적하는 columnList의 각 칼럼을 위해 호출된다. 이 재귀 동작은 단계(224)에서 그룹내의 모든 칼럼이 처리될 때까지 발생한다.

이제 도 9B를 참조하면, 일단 그룹이 아닌 칼럼이 발견되면, 컴퓨터 시스템은 칼럼이 유형 DataColumn 또는 유형 KeyColumn인 지의 여부를 판정한다(단계(228)). 만일 칼럼이 이 두 유형중의 하나로 정의되면, 처리는 단계(232)로 가고 여기서 현재 스키마의 칼럼이 제안된 스키마에서 발견되는 지의 여부가 판정된다. 이것은 동일한 Col_Id번호를 갖는 칼럼에 대해 제안된 테이블에서 정의된 columnList를 탐색분석하므로써 분석된다. 칼럼이 제안된 테이블에서 발견되지 않는다면, 사용자는 칼럼이 고려중인 테이블로부터 삭제되게 하는 객체 모델 변경을 수행한다. 칼럼이 테이블로부터 삭제되기 이전에, 컴퓨터 시스템은 컴퓨터 시스템이 다른 테이블로 이동되어야 할 데이터를 포함하는 지의 여부를 판정해야 한다. 이 프로세스는 도 13에 나타나 있고 아래에서 설명된다. 도 13에 나타난 단계를 수행한 후, 단계(236)에서 칼럼은 고려중인 현재 테이블로부터 삭제되도록 준비된다.

아래에서 더욱 설명되는 바와 같이, 대응 데이터베이스에 대해 행해진 임의의 변경은 동시적으로 이루어지지 않는다. 데이터베이스 테이블에 대해 행해진 모든 변경은 현재 및 제안된 스키마가 완전히 비교될 때 까지 저장된다. 이때, 데이터베이스의 테이블에 대한 변경이 행해진다.

만일 칼럼이 현재 스키마의 테이블 및 제안된 스키마의 테이블 모두에서 발견된다면, 단계(240)에서 컴퓨터 시스템은 dataType, logicalLength, mullAllowed, columnName, dataLength 및 scale 특성이 제안된 스키마에서와 같은 현재 스키마에서와 동일한지의 여부를 판정한다. 만일, 그 결과가 아니오 이면, 컴퓨터 시스템의 특성은 단계(244)에서 갱신된다.

단계(248)(도 9C)에서, 컴퓨터 시스템은 분석되는 칼럼이 유형 ForeignKey 인지의 여부를 판정한다. 만일, 그 결과가 예이면, 컴퓨터 시스템은 그후 키가 제안된 스키마에 존재하는 지의 여부를 판정한다(단계(248)). 만일 키가 제안된 스키마에 존재하지 않는다면, 컴퓨터 시스템은 컴퓨터 시스템이 박스254로 표시된 도 13에 나타난 단계들을 이용하여 데이터베이스의 다른 테이블로 이동되어야 할 데이터를 포함하는 지의 여부를 판정한다. 단계(256)에서, 키는 그후 현재 데이터베이스의 테이블로부터 탈락된(dropped) 것으로 표기된다.

만일 키가 현재 스키마 및 제안된 스키마의 테이블 모두에 존재한다면, 컴퓨터 시스템은 단계(240 및 244)에서 설명된 것과 마찬가지의 방식으로 단계(260)에서 외부 키를 정의하는 멤버 키 칼럼을 비교 및 갱신한다.

단계(264)에서, 컴퓨터 시스템은 고려중인 테이블에 대한 columnList의 모든 칼럼이 분석되었는 지의 여부를 판정한다. 만일, 그 결과가 아니오이면, 컴퓨터 시스템은 단계(216)로 가고(도 9A) 현재 테이블에 대한 columnList의 다음 엔트리가 분석된다.

일단 현재 스키마내의 테이블에 포함된 모든 칼럼이 분석되었으면, 단계(268)(도 9D)에서 컴퓨터 시스템은 현재 테이블에서 발견되지 않는 제안된 테이블에 존재하는 임의의 칼럼이 있는 지의 여부를 판정한다. 만일 그 결과가 예 이면, 단계(272)에서 추가의 칼럼이 고려중인 데이터베이스에 추가된다.

현재 스키마내의 테이블의 모든 칼럼이 분석된 후, 단계(276)에서 컴퓨터 시스템은 루프를 시작하고 특정 테이블에 대해 유지된 indexList내의 각 인덱스를 분석한다. 단계(280)에서, 현재 스키마내의 indexList의 엔트리가 발견도고 그것이 제안된 스키마에 대한 대응 엔트리와 동일한 것인 지의 여부를 판정한다. 만일 현재 스키마내의 인덱스가 제안된 스키마에서 발견되지 않으면, 그것은 탈락된다. 만일 동일 하지 않으면, 단계(284)에서 indexList의 엔트리가 갱신된다. 단계(288)에서, indexList의 모든 엔트리가 분석되었는 지의 여부를 판정한다. 만일 그 결과가 아니오 이면, 처리는 단계(276)로 되돌아가고 인덱스의 리스트의 다음 엔트리가 분석된다. 단계(288) 후에, 현재 테이블에서 발견되지 않은 제안된 테이블의 나머지 인덱스가 생성된다.

단계(292)(도 9E)에서, 컴퓨터 시스템은 현재 스키마의 주키가 제안된 스키마의 주키와 동일한 지의 여부를 판정한다. 만일 그 결과가 아니오 이면, 현재 데이터베이스의 주키는 단계(296)에서 갱신된다. 만일 새로운 주 키에 대한 데이터가 자격을 갖추지 못한다면 즉, 널 값을 포함하거나 고유하지 않으면, 에러 상태가 나타난다.

컴퓨터 시스템은 따라서 단계(300)에서 현재 스키마의 테이블의 tableName 특성이 제안된 스키마에 정의된 tableName 특성과 동일 한 지의 여부를 판정한다. 만일 그 결과가 아니오 이면, 단계(304)에서 현재 데이터베이스의 테이블의 이름 특성이 갱신된다.

단계(308)에서, 컴퓨터 시스템은 tableList의 모든 엔트리가 분석되었는 지의 여부를 판정한다. 만일 그 결과가 아니오 이면, 단계(204)로 되돌아가고(도 9A), 다음 테이블이 상기와 같은 방식으로 분석된다. 일단 모든 테이블이 분석되면, 컴퓨터 시스템은 단계(312)(도 9F)에서 현재 스키마에 있지 않은 제안된 스키마에 임의의 테이블이 존재하는 지의 여부를 판정한다. 만일 그 결과가 예 이면, 단계(316)에서 추가 테이블이 현재 데이터베이스에 추가된다.

상기한 바와 같이, 현재 데이터베이스내의 관계형 테이블에 행해진 변경이 수행되어야 할 액션 리스트에 저장된다. 객체 모델내의 성분에서의 변경 결과로서 수행되어야 할 각각의 액션은 다음 부류의 사례에 의해 유지된 외부 키 동작, 인덱스 동작 또는 칼럼 동작 리스트에 추가된다.

```
부류 SPTable
//     테이블 동작 정의등
Create();
Drop();
ChangeName(oldName);
//     columnOperationList에 엔트리를 추가하는 칼럼동작 정의
AddColumn(pColumn);
DropColumn(currentColumnName);
ChangeColumnName(pNewCOl, oldColumnName);
ChangeColumnConstraint(pColumn);
ChangeColumnDataType(pColumn);
//     indexOperationList에 엔트리를 추가하는 인덱스동작 정의
AddIndex(pIndex);
DropIndex(currentIndexName);
ChangeIndexConstraint(pIndex);
//     foreignkeyOperationList에 엔트리를 추가하는 키동작 정의
ChangePrimaryKey( );
AddForeignKey(pForeignKey);
DropForeignKey(currentForeignKeyName);
//     테이블 기능 생성/변경      //실행 유형
CreateIt();               //CREATE
DropIt();                 //DROP
Alter0();                 //ALTER
Alter1();                 //REPLACE_NEWNAME
Alter2();                 //REPLACE_SAVEDATA
AddIndex();               //
DropIndex();              //
AddForeignKey();          //
DropForeignKey();         //
//     베이스 SPTable 데이터 포맷
oldName;                  //데이터베이스에 정의된 최후테이블        이름
columnOperationList;      //컬럼에 대한 변경동작 리스트
indexOperationList;       //인덱스에 대한 변경동작 리스트
foreignkeyOperationList;  //외부 키에 대한 변경동작 리스트
ExecutionType;            //테이블 변경동작의 현재 모드
*pTableDef;               //새로운 테이블 정의 객체로 포인터
*pDMTDef;                 //임시 테이블 정의로 포인터
};
```

부류 정의로부터 알 수 있는 바와 같이,부류 SPTable은 일련의 칼럼, 인덱스, 및 키 정의 방법을 포함한다. 각각의 방법은 적절한 데이터를 각각 column Operation- List, indexOperationList 및 foreignkeyOperationList에 추가한다. 또한, 부류는 컴퓨터 시스템이 테이블내의 칼럼을 재정의하는 수정을 수행할 뿐만 아니라 테이블 탈락 또는 테이블 생성과 같은 데이터베이스 수정을 할 수 있는 여러 생성/변경 방법을 포함한다. 이들 방법을 구현하기 위해 필요한 특정 동작은 컴퓨터 시스템이 기초로 되는 관계형 데이터베이스를 동작 또는 제어 하기위해 사용하는 데이터베이스 관리 시스템에 종속하여 변할 수 있는 일련의 SQL문을 필요로 한다. 예를들어, 컴퓨터는 관계형 데이터베이스를 동작시키기 위해임의의 여러 관계형 데이터베이스 프로그램 또는 Borland Paradox® 또는 Microsoft Access® 를 이용할 수 있다. 본 발명에서, 사용자에게는 데이터베이스 프로그래밍 유형을 식별할 것이 요구되고 컴퓨터는 이에따라 소망 타스크를 수행하기 위해 적절한 루틴을 선택한다.

모든 데이터베이스 시스템이 동일 방식으로 테이블을 다루는 SQL 명령을 지지하지는 않는다. 예를들어, 어떤 데이터베이스 명령어는 일단 테이블이 생성되면 테이블의 이름의 변경을 허용하지 않는다. 그러므로 본 발명은 기초가 되는 관계형 데이터베이스를 제어하는 데이터베이스 관리 시스템의 성능에 의거하여 동작을 수행하기 위해 선택된 특정 SQL 명령을 조정해야 한다. 이 방법에서 임의의 특정 데이터베이스에 대해 수행되어야 하는 동작은 일단 컴퓨터 시스템이 사용자가 사용하는 데이터베이스가 어느 것인지를 알게 되면 메모리로 로드되는 맵 데이터 구조에 리스트된다. 다음 테이블은 동작을 수행하는 여러 방법중의 하나로 관계형 데이터베이스 테이블에 행해져야 할 변경을 관련시키는 맵의 일예이다.

[표 5]

동작 유형	실행 유형
새로운 테이블 생성	CREATE
현재 테이블 탈락	DROP
테이블 이름 변경	REPLACE_NEWNAME
새로운 칼럼 추가	ALTER
새로운 요구된 칼럼 추가	REPLACE_SAVEDATA
현재 컬럼 탈락	ALTER
컬럼 이름 변경	ALTER
칼럼 제약조건 변경	REPLACE_SAVEDATA
데이터 유형 변경	ALTER
새로운 인덱스 추가	ALTER
현재 인덱스 탈락	ALTER
인덱스 제약조건 변경	ALTER
새로운 외부 키 추가	REPLACE_SAVEDATA
현재 외부 키 탈락	REPLACE_SAVEDATA
인덱스 외부 키 변경	REPLACE_SAVEDATA
데이터 이동	REPLACE_SAVEDATA

표 5는 SPTable 부류에 정의된 5개의 상이한 생성/변경 테이블 방법중의 하나에 소망 관계형 테이블 변경을 매핑시킨다. 이 방법은 CREATE, DROP, ALTER, REPLACE_NEWNAME, REPLACE_SAVEDATA로 레이블링 된다. 각각의 방법은 특정 데이터베이스에 대해 정의된다. 예를들어, Microsoft Access® 로 생성된 데이터베이스와 함께 사용될 REPLACE_NEWNAME 루틴은 Borland Paradox® 로 생성된 데이터베이스에 대해 상이하게 정의될 수 있다.

데이터베이스내의 테이블을 변경하기위해, 컴퓨터 시스템은 테이블 수정 방법중의 하나를 선택하고 소망 동작을 테이블을 위해 수행되어야 할 동작 리스트에 추가한다. 예를들어, 기존 데이터베이스에 테이블에 칼럼을 추가하기 위해, 컴퓨터 시스템은 방법 SPTable::AddColumn(pColumn)을 호출한다. 이 방법은 컴퓨터 시스템에게 포인터 인수에 의해 참조된 칼럼에 추가하기 위해 columnOperationList에 엔트리를 추가한다. 또한 방법 SPTable::AddColumn(pColumn)은 ExecutionType 멤버 변수를 ALTER와 동일하게 설정하기위해 표5를 이용할 것이다. ExecutionType은 또다른 동작이 ExecutionType 변수를 더욱 고 순위인 다른 것으로 갱신할 때 까지 ALTER로 남아있는다. 예를들어, 방법 SPTable::ChangeColumnConstraint(pColumn)은 ExecutionType을 REPLACE_SAVEDATA로 설정한다. 본 실시예에서, 최고순위 실행 유형은 REPLACE_NEWNAME, ALTER, DROP 및 CREATE가 순서대로 뒤따르는 REPLACE_SAVEDATA이다.

상용 관계형 데이터베이스 관리 시스템의 대부분은 관계형 데이터베이스 테이블을 생성 및 탈락시키기 위해 표준화된 SQL 메커니즘을 제공한다. 따라서, CreateIt() 및 DropIt() 방법은 상이한 데이터베이스 유형간에 크게 변하지 않는다. 그러나, ALTER, REPLACE_NEWNAME, 및 REPLACE_SAVEDATA 루틴에 대응하는 방법은 상이한 데이터베이스 관리 시스템간에 변화할 것이다.

ALTER 방법은 매우 정교한 데이터베이스 변경을 지지하는 데이터베이스를 위해 채용된다. 데이터베이스는 칼럼을 기존 테이블에 추가하고, 데이터베이스의 칼럼 및/또는 테이블을 개명하고 그리고 전체 테이블을 재정의하지 않고 테이블의 키(주키 및 외부 키)를 재정의 하기위해 SQL문을 지지한다.

REPLACE_NEWNAME 방법은 전체 테이블을 재정의하지 않고 테이블이 개명될 수 있게 하는 데이터베이스 관리 시스템을 위해 사용된다.

도 10A는 관계형 데이터베이스없이 테이블을 완전히 재정의하기 위한 REPLACE_NEWNAME 실행 방법을 사용할 때 컴퓨터 시스템에 의해 수행된 단계를 개괄한다. 단계(320)에서, 새로운 데이터베이스 테이블이 새로운 이름으로 생성된다. 단계(324)에서, 구 테이블로 부터의 데이터는 새로운 테이블로 복사된다. 단계(328)에서, 구 테이블은 데이터베이스로부터 탈락된다.

만일 데이터베이스 관리 시스템이 임의 동작을 완료하기 위해 단순한 SQL문을 사용하여 지지하지 않는다면, 더욱 정교한 방법이 사용되어야 한다. 도 10B는 REPLACE_SAVEDATA 실행 방법에 수행된 단계를 개괄한다. 단계(332)에서, 제안된 테이블과 동일 정의를 갖는 임시 테이블이 임의의 키 제약조건 없이 데이터베이스에서 생성된다. 단계(336)에서, 데이터는 기존 테이블로부터 임시 테이블로 이동된다. 본래 테이블은 단계(340)에서 데이터베이스로부터 탈락된다. 단계(344)에서, 새로운 테이블이 데이터베이스에 생성되고 단계(348)에서 데이터는 임시 테이블로부터 새로운 테이블로 복사된다. 마지막으로 단계(352)에서 임시 테이블은 데이터베이스로부터 삭제된다.

알 수 있는 바와 같이, 표 5에 나타나고 상기한 맵은 기초가 되는 관계형 데이터베이스를 제어하는 특정 데이터베이스 관리 프로그램에 의해 지지된 SQL특징에 종속하여, 테이블을 수정하기 위해 사용되어야 할 실행 루틴을 결정한다.

데이터베이스키마내에서 데이터 무결성을 보존하기 위해, 테이블이 수정되어야 할 순서가 중요하다. 본 발명의 본 실시예에서, 다른 테이블에서 외부 키에 의해 참조된 주키를 갖는 케은 외부 키를 포함하는 테이블 이전에 수정되어야 한다. 상기한 바와 같이, 데이터베이스에서 각 테이블을 수정하기위해 필요한 동작은 SPTable 부류의 사례로서 저장된다. 이들 사례는 데이터 무결성을 보존하기 위해 올바른 순서로 리스트에 위치된다.

도 11A는 SPTable 부류의 사례에 대응하는 다수의 엔트리를 포함하는 Change TableList(380)의 표현을 도시한다. 각각의 엔트리는 테이블(382,384,386 및388)에 대응한다. 테이블(382)은 주키를 포함하지만 기타 테이블을 참조하진 않는다. 테이블(384)은 테이블(388)에 대한 외부 키를 포함하며 테이블(388)은 테이블(386)에 대한 외부 키를 포함한다. 데이터 무결성 제약조건을 위반하지 않기 위해, 기타 테이블의 외부 키에 의해참조된 주키를 갖는 테이블은 외부 키를 포함하는 테이블 이전에 수정되어야 한다. 이러한 순서 체계는 최소 종속 순서로 참조된다.

본 발명에서, SPTable 부류의 사례는 최소 종속 순서의 리스트에 삽입된다. 기타 테이블에 대해 관계를 갖지않는 테이블은 리스트의 아무곳에 삽입될 수 있고, 외부 키를 포함하는 테이블은 그들이 참조하는 테이블 이후에 참조되어야 한다. 도 11A에 도시된 예에서, 테이블(386)은 반드시 테이블(388)이전에 수정되어야 하고, 테이블(388)은 반드시 테이블(384)이전에 수정되어야 한다. 테이블(382)은 어느 때고 수정될 수 있다.

도 11B는 SPTable 부류의 사례를 최소 종속 순서로 ChangeTableList로 호칭되는 리스트에 삽입하기위해 취해지는 단계를 도시한다. 컴퓨터는 단계(400)에서 ChangeTableList에 삽입되어야 할 각각의 SPTable 부류의 사례를 분석하기 시작한다. 단계(402)에서, 컴퓨터 시스템은 이미 ChangeTableList에 있는 각각의 엔트리를 분석하는 루프를 시작한다. 단계(404)에서, 컴퓨터 시스템은 이미 리스트에 있는 엔트리가 추가되어야 할 엔트리에 대한 액티브 외부 키를 포함하는 지의 여부를 판정한다. 만일, 그 결과가 아니오 이면, 컴퓨터 시스템은 단계(406)에서 이미 리스트에 있는 엔트리가 분석되었는 지의 여부를 판정한다. 만일, 단계(406)에서의 결과가 아니오 이면, 컴퓨터 시스템은 단계(402)로 되돌아가고 ChangeTableList에 있는 다음 엔트리가 분석되어 진다. 만일, 단계(406)에서의 결과가 예 이면, 단계(408)에서 새로운 엔트리가 리스트의 끝에 추가된다.

만일, 단계(404)에서의 결과가 예 이면, 이미 리스트에 있는 엔트리는 추가되어야 할 엔트리에 대한 액티브 외부 키를 포함하고, 단계(410)에서 새로운 외부 키를 포함하는 엔트리 이전에 새로운 엔트리가 추가된다.

단계(412)에서, 컴퓨터는 새롭게 추가된 엔트리 이후에 존재하는 각각의 엔트리를 분석한다. 단계(414)에서, 컴퓨터 시스템은 새롭게 추가된 엔트리가 리스트에 새롭게 엔트리가 추가된 후 위치된 엔트리에 대해 외부 키를 포함하는 지의 여부를 판정한다. 만일, 그 결과가 예 이면, 단계(416)에서 컴퓨터 시스템은 새로운 엔트리에 의해 참조된 엔트리를 외부 키를 포함하는 새로운 엔트리 이전에 위치로 이동시킨다. 만일, 단계(414)에서의 결과가 아니오 이면, 단계(418)에서 컴퓨터 시스템은 ChangeTableList에 남아있는 모든 엔트리가 분석되었는 지의 여부를 판정한다. 만일, 그 결과가 아니오 이면, 컴퓨터 시스템은 단계(412)로 되돌아 간다. 리스트의 끝에 도달하면, 단계(420)에서 컴퓨터 시스템은 SPTable 부류의 모든 사례가 리스트에 추가되었는 지의 여부를 판정한다. 만일, 그 결과가 아니오 이면, 컴퓨터 시스템은 단계(400)로 되돌아 가도록 다음 새로운 엔트리를 리스트에 추가한다.

일단 SPTable 부류의 모든 사례가 ChangeTableList에 추가되었으면, Change TableList는 단계(422)에서 완료되는 것으로 지시된다. 도 11B에 도시된 단계를 수행한 후, 데이터베이스의 테이블은 데이터 무결성을 보존하는 순서로 수정된다.

상기한 순서화 체계에 대한 유일한 예외는 테이블 셋트가 서로에 대해 원형 종속 관계를 가질 때이다. 도 11C는 3개의 테이블(T_1,T_2,T_3) 가운데 이러한 유형 관계를 예시한다. 테이블을 재정의하지 않고 외부 키를 탈락 또는 추가하기 위해 단순 SQL 신택스를 통해 관계형 테이블에 대한 외부 키의 추가 또는 탈락을 특정 데이터베이스 시스템이 지원할 때, 다른 테이블 정의를 수정하기 전에 영향을 받은 외부 키 정의를 제일 먼저 탈락시키고 그후 모든 필요한 관계를 재저장하므로써 임의의 순환관계가 취급될 수 있다. 외부 키 관계를 탈락시킨 후 후, 테이블에서 수행되어야 할 동작을 나타내는 SPTable 부류의 사례가 임의의시점에서 Change TableList 에 삽입될 수 있다. 이 선택사항이 이용불가할 때, 테이블은 명시되지 않은 순환 관계를 정의하는 임의의 순서 및 임의의 관계로 수정될 수 있다. 영향을 받은 테이블은 입력되어야 할 데이터를 저장하기 위해 대응 키 칼럼을 갖도록 정의되지만, 칼럼은 외부 키로서 명시되지 않는다.

데이터베이스에서 수정되는 다수의 테이블은 이전에 저장된 데이터를 포함한다. 그러므로, 대응 객체 모델에 대해 사용자가 행한 변경을 정확히 반영하기위해 일 테이블로부터 다른 테이블로 데이터가 이동되어야 하는 체계 수정이 있다. 이것을 데이터 이동이라 한다.

전형적으로, 데이터 이동의 사례는 데이터 칼럼이 테이블로부터 탈락되었을 때 지시된다. 따라서, 도 9B의 단계(236) 이전에, 컴퓨터 시스템데이터를 포함하는 테이블로부터 칼럼이 삭제되어야 하는 지의 여부를 판정한다. FoeignKey 칼럼이 데이터베이스로부터 탈락되어야 할 때, 단계(256)에서 컴퓨터는 칼럼이 탈락되기 이전에 칼럼이 데이터를 포함하는 지의 여부를 판정한다.

컴퓨터 시스템이 일 관계형 테이블로부터 다른 테이블로 데이터를 이동시키는 방법을 충분히 이해하기 위해, 도 12는 데이터 이동에 필요한 대응 관계형 테이블 및 객체 모델에 대한 표현을 나타낸다. 객체 모델은 사무실,고용인 및 관리자를 각각 나타내는 3개의 객체(500,510 및 520)를 포한한다. 객체(500)는 관계형 테이블(530)과 연관된다. 객체(510)는 관계형 테이블(540)과 연관되고 객체(530)는 관계형 테이블(550)과 연관된다.

도시된 예에서, 사용자는 객체(500)로부터 '전화번호'로 레이블링 된 단순 값 성분을 이동시켜서 객체(510)에 위치시킨다. 마찬가지로, '이름'으로 레이블링 된 단순 값 객체를 객체(520)로부터 객체(510)로 이동시킨다. 성분의 각각은 대응 테이블(530 및 550)에 저장된 데이터와 연관된다.

객체 모델 변경을 나타내기 위해, 데이터는 두 출발지 테이블(530 및 550)로부터 목적지 테이블(540)로 이동되어야 한다. 이것은 목적지 테이블(즉, 테이블(540)로부터 '사회_보장_번호'로 레이블링 된 칼럼에 저장된 데이터)에 저장될 수 있는 임의의 기존 데이터뿐만 아니라, 출발지 테이블(530 및550)로부터 '전화번호' 및 '이름'으로 레이블링 된 칼럼에 저장된 데이터를 수용하는 임시 테이블(도시되지 않음)을 생성하므로써 달성된다. 데이터가 임시 테이블로 이동된 후, 테이블(530 및 550)의 '전화번호' 및 '이름'으로 레이블링 된 칼럼은데이터베이스로부터 삭제될 수 있다. 본래의 테이블(540)은 탈락되고, 새로운 테이블(540')이 생성되고 임시 테이블로부터의 데이터는 새로운 테이블(540')로 복사된다. 마지막으로 임시 테이블이 삭제된다.

도 13은 기존 데이터베이스의 일 테이블로부터 다른 테이블로 데이터를 이동시키기 위해 본 발명의 컴퓨터 시스템에 수행된 단계를 도시한다. 단계(450)에서 시작하여, 컴퓨터 시스템은 컴퓨터 시스템이 데이터를 포함하는 테이블로부터 삭제되어야 하는 지의 여부를 판정한다. 당업자에 의해 인식될 수 있는 바와 같이, 이것은 다수의 행을 칼럼으로 복귀시킬 수 있는 SQL문을 사용하여 수행된다. 만일 컬럼이 데이터를 포함한다면, 컴퓨터 시스템은 단계(452)에서 동일한 Col_Id번호를 갖는 칼럼을 위해 삭제되어야 할 칼럼을 포함하는 테이블에 대한 임의의 부모 테이블을 탐색한다.

단계(456)에서, 컴퓨터 시스템은 삭제되어야 할 칼럼으로서 Col_Id번호를 갖는 칼럼이 발견되었는 지의 여부를 판정한다. 만일 단계(456)에서의 결과가 아니오 이고 Col_Id가 발견되지 않았다면, 컴퓨터 시스템은 칼럼이 탈락되어야 할 테이블에 대한 외부 키에 의해 관련되는 객체 테이블 탐색을 시작한다(단계(458)). 단계(460)에서, 삭제되어야 할 칼럼으로서 Col_Id번호를 갖는 칼럼이 관련 테이블에서 발견되었는 지의 여부를 판정한다.

일단 삭제되어야 할 칼럼이 데이터베이스의 다른 칼럼에서 발견되었다면, 컴퓨터 시스템은 단계(462)에서 DMData 또는 DMForeignData 부류중의 하나를 발생시킨다.

```
부류  DMData
{
SourceTable;        //데이터가 오는 곳
DestTable;          //데이터를 수용하기 위한 테이블
SourceCol;          //데이터를 제공하기 위한 컬럼
DestCol;            //데이터를 수용하기 위한 컬럼
MigrationType;      //일-대-일, 일-대-다수, 등
DataType;           //값 또는 링크
Migrationpath;      //출발지와 목적지 테이블간의 외부 키 관련 리스트
};
```

DMData 부류는 일 테이블로부터 다른 테이블로 삭제되어야 할 데이터를 포함하는 칼럼에 대한 정보를 저장하기위해 사용된다. 이동 경로는 목적지 테이블과 출발지 테이블을 관련시키는 일련의 테이블을 상세한다. 이동 경로는 리스트의 다음 테이블과 관련한 외부 키 및 각각의 테이블에 대한 데이터 구조의 리스트이다. 이동 경로를 계산 하기위해 컴퓨터 시스템에 의해 수행된 단계는 아래에서 더욱 상세히 설명된다.

이동되어야 할 칼럼이 ForeignKey 유형이면, DMForeignData 부류가 생성된다.

```
부류 DMForeignData: 공용 DMData
{
*pDestForeignKey;          //제안된 테이블의 외부키에 대한 포인터
*pSourceForeignKey;        //현재 테이블의 외부키에 대한 포인터
};
```

도 12에 도시된 바와 같이, 테이블은 다중 출발지 테이블로부터 데이터를 수용할 수 있다. 따라서, 다음 부류는 목적지 테이블로 이동되어져야 할 데이터를 제공하는 각각의 출발지 테이블을 설명하는 DMForeignData 및 DMData 의 사례를 저장한다.

```
부류  DMTable
{
DMDataList;         //DMForeignData 또는 DMData의 모든 사례를 리스팅
*TableDef;          //데이터를 수용하기 위해 목적지 테이블의 정의에
//대한 포인터
TempTableName;      //전송 동안 데이터를 보유하기 위한 임시 테이블
};
```

DMTable의 단 하나의 사례만이 임의의 목적지 테이블에 대해 생성된다. 따라서 단계(464)(도 13)에서, 컴퓨터 시스템은 이 부류의 사례가 목적지 테이블을 위해 생성되었는 지의 여부를 판정하고 만일 그 결과가 아니오 이면 그 테이블의 사례를 생성한다.

단계(466)에서, 컴퓨터는 목적지 테이블로 이동되어야 할 데이터의 각 칼럼에 대해 DMDataList로 DMForeignData 또는 DMData의 사례를 추가한다. 만일 목적지 테이블이 본래의 테이블에서 발견된 데이터를 포함하여야 하면, DMData의 사례는 새로운 테이블로 복사되어야 할 데이터의 각각의 본래의 테이블을 위해 생성된다.

단계(468)에서, 컴퓨터는 목적지 테이블을 위해 정의된 SPTable 부류의 사례를 생성한다. 상기한 바와 같이, SPTable 부류는 테이블에서 수행되어야 할 사례를 발견 또는 생성한다. SPTable 부류의 사례는 목적지 테이블을 위해 생성된 DMTable의 사례에 대한 포인터를 포함한다. 데이터를 수용하는 임의의 테이블에 대한 실행 유형은 도 10B에 개괄되고 상기된 테이블 수정 단계를 이용하기 위해 항상 REPLACE_SAVEDATA로 설정된다.

다음은 도 12에 도시된 예를 위해 3개의 상이한 테이블로부터 데이터로 삽입하고 임시 테이블을 생성하는 SQL문의 기본 포맷을 나타낸다.

```
CREATE TABLE        TempTbl(사회_보장_번호,Int,이름 텍스트(25),
전화 텍스트(25))
INSERT INTO         TempTbl(사회_보장_번호,Int,전화번호)
SELECT               고용인.사회_보장_번호,관리자.이름,사무실.전화
FROM                 고용인,관리자,사무실.
WHERE                관리자.사회_보장_번호,는 고용인.사회_보장_번호에
대한 외부키이다
AND                  사무실.타이틀은 관리자.타이틀에 대한 외부키이다
```

본 표준 SQL을 이용하여, 상이한 테이블로부터 순차로 다중 데이터 셋트를 삽입하는 것은 불가능하지 않다. 상기 SQL문에서 나타나는 바와 같이, 모든 3개의 출발지 칼럼 및 이들의 대응 테이블은 동시에 상세되어야 한다. WHERE절은 상이한 테이블로 부터의 데이터 행이 각각의 데이터 셋트의 특정 이동 경로에 따라 어떻게 관련되는 지를 상세한다.

표 6은 데이터 삽입에 대해 적절하게 테이블을 정의하기 위한 상이한 이동 경로 유형을 분류한다. 이 테이블은 값(즉, 외부 키가 아닌)으로서 데이터 이동을 위한 것이다. 값 유형 이동 경로에 대한 5개의 상이한 이동 경로 유형이 있다.

[표 6]

값 유형 데이터 이동

일_대_일	다음 결과로서 카디널리티 변경없이 이동된 데이터: --단일 일_대_일 객체 링크 관계를 통해 관련된 객체로 성분을 이동
일_대_다수	다음 결과로서 카디널리티 증가로 이동된 데이터: -- 1로부터 N으로 변경된 성분 카디널리티 -- 단일값을 갖는 성분을 다중값을 갖는 그룹 성분으로이동
다수_대_일	다음 결과로서 카디널리티 감소로 이동된 데이터: -- N으로부터 1로 변경된 성분 카디널리티 --다중값을 갖는 그룹으로부터 그굴 또는 객체를 포함하는 성분을 이동
다수_대_다수	다음 결과로서 데이터의 또다른 셋트로 병합된 데이터: -- 다중값을 갖는 성분을 단일값을 갖는 성분으로 변경하고 다중값을 갖는 그룹으로 이동
복사	본래 테이블로부터 임시/새로운 테이블로 단순 데이터를 복사

일-대-일 값 유형 데이터 이동의 예가 도 12에 도시되어 있고, 여기서 이름 성분은 관리자 객체(520)로부터 고용인 객체(510)로 이동된다. 이들 두 객체는 단일 값을 갖는 객체 링크 성분에 의해 관련된다.

일-대-일 링크 유형 데이터 이동의 예가 도 7에 도시되어 있고, 여기서 전공 성분의 카디널리티는 1로부터 N으로 변경된다. 다수-대-일 값 유형 데이터 이동은 도 7에 도시된 예의 역이다.

다수-대-일 값 유형 데이터 이동의 예가 도 12A에 도시되어 있다. 여기서 객체(550)는 학생을 나타낸다. 객체(550)는 '학생_ID' 및 '이름'으로 레이블링 된 두 개의 단일 값을 갖는, 단순 값 성분을 포함한다. 또한 객체(550)는 '전공'로 레이블링 된 다중값을 갖는, 단순 값 성분과 '년도데이터'로 레이블링 된다중값을 갖는 그룹 성분을 포함한다.

테이블(552)은 데이터베이스의 객체(550)을 표현한다. 테이블(554)은 대용 키, 부모 테이블(552)에 대한 외부 키를 보유하는 컬럼과, 연도와 지피에이에 대한 데이터를 저장하는 칼럼을 포함한다. 테이블(556)은 대용 키, 부모 테이블(552)에 대한 외부 키를 보유하는 컬럼과 학생의 전공을 저장하기 위한 칼럼을포함한다.

다수-대-다수 값 유형 데이터 이동은 사용자가 '전공'로 레이블링 된 성분의 카디널리티를 변경하면 발생되고, 그것을 다중값을 갖는 그룹 '년도데이터'로 이동시킨다. 데이터베이스에서 이러한 변경을 표현하기 위해, 학생의 전공을 저장하는 칼럼은 테이블(556)로부터 인척 테이블(554)로 이동된다.

복사 유형 데이터 이동은 데이터가 목적지 테이블의 본래 버전으로부터 목적지 테이블의 수정된 버전으로 이동될 때 발생한다. 예를들어, 테이블(540)의 사회_보장_번호의 데이터를 도 12에 도시된 테이블(540')로 이동하는 것은 복사 유형 데이터 이동이다.

또다른 이동 유형 셋트는 일 테이블로부터 다른 테이블로 외부 키를 표현하는 데이터 이동을 분류하기 위해 사용된다. 두 객체간의 관계가 변경될 때, 대응 외부 키에 대해 몇몇 변경이 가해져야 할 필요가 있다. 표 7은 링크 유형 데이터 이동의 다양한 유형을 분류한다.

[표 7]

링크 유형 데이터 이동

일_대_일	참조된 객체 테이블로 이동된 외부 키
다수_대_일	성분 테이블(즉, 다중값을 갖는 그룹의 링크)로부터 참조된 객체 테이블로 이동된 외부 키
갱 신	대응 주 키 칼럼의 변경에 따라 변경된 외부 키 칼럼 정의, 즉 관련 데이터에서 어떤 변경도 없음
대 체	부모 테이블의 변경에 따라 정의된 새로운 외부 키

일-대-일 링크 유형 데이터 이동의 예가 도 8에 도시되어 있고, 여기서 외부 키 칼럼은 교수 테이블(182)로부터 학생 테이블(186)로 이동된다.

다수-대-일 링크 유형 데이터 이동의 예가 도 12B에 도시되어 있다. 여기서 객체(570)은 학생을 표현한다. 이 객체는 3개의 성분 즉, '학생_ID' 및 '이름'으로 레이블링 된 두 개의 단일 값을 갖는 성분뿐만 아니라 '년도데이터'로 레이블링 된 그룹 성분을 포함한다. 그룹 성분은 '년도'로 레이블링 된 단순 값 성분과 '튜터'로 레이블링 된 객체 링크 성분을 포함한다. 객체 링크 성분은 학생 객체(550)를 학생의 튜터를 표현하는 객체(552)에 링크시킨다. 다수-대-일 링크 유형 데이터 이동은 사용자가 다중값을 갖는 그룹 연도데이터로부터 튜터 객체 성분을 이동시킬 때 발생한다.

수정 이전에, 도 12B에 도시된 객체 모델은 4개의 테이블에 의해 데이터베이스에서 표현된다. 테이블(574)은 학생의 식별 번호 및 학생의 이름을 보유하는 칼럼을 갖는다. 테이블(576)은 테이블(574)의 자식 테이블이다. 테이블(576)은 대용 키, 테이블(574)에 대한 외부 키 및 연도를 보유하는 3개의 칼럼을 갖는다. 테이블(578)은 테이블(576)의 자식 테이블이다. 테이블(578)은 대용 키, 테이블(574)에 대한 외부 키, 및 튜터를 설명하는 데이터를 보유하는 테이블(580)에 대한 외부 키를 보유하는 칼럼을 갖는다. 다중값을 갖는 갖는 그룹 '년도데이터'로부터 튜터 객체 성분을제거하므로써, 컴퓨터 시스템은 테이블(578)을 삭제하고 테이블(580)과 테이블(574)을 링크시키기 위한 외부 키를 보유할 테이블(580)에 칼럼을 추가한다.

링크 유형 데이터 이동의 갱신 및 교체는 테이블의 주키가 외부 키로서 주키 값을 사용하는 테이블의 대응 변경을 야기할때 발생한다.

테이블로부터 탈락되어야 할 칼럼이 제안된 스키마의 다른 테이블에서 발견될 때, 이동 데이터 유형(값 또는 링크)이 식별된다. 이 정보에 의거하여, 데이터를 이동시킬 SQL문을 작성하기 위해 필요한 이동 경로는 목적지 테이블을 위해 생성된 DMTable 부류 사례의 각각의 DMData 및 DMForeignData를 위해 산출된다.

도 14A 및 14B는 값 유형 데이터 이동을 위한 이동 경로 및 링크 유형 데이터 이동을 계산 하기위해 컴퓨터 시스템에 의해 수행된 단계를 나타낸다. 도 14A를 참조하면, 컴퓨터는 먼저 이동 경로 유형(즉 일-대-일, 일-대-다수, 다수-대-일, 다수-대-다수 또는 복사)을 검사하여 '값'I과 동일한 DataType를 갖는 DMData 또는 DMForeignData의 각각의 사례를 분석하는 루프를 단계(700)에서 시작한다. 단계(702)에서 소스 테이블은 그후 계산되는 이동 경로에 제 1 엔트리로서 추가된다.

단계(704)에서, 컴퓨터는 이동 유형이 일-대-일인 지의 여부를 판정한다. 만일 그 결과가 예이면, 단계(706)에서 제안된 스키마의 목적지 테이블의 테이블_Id를 매치시키는 테이블_Id를 갖는 현재의 스키마에서 관련된 객체 테이블을 발견한다. 도 12에 도시된 일-대-일 예에 대해, 컴퓨터는 고용인 테이블(540)을 위치시킨다. 다음에, 컴퓨터 시스템은 단계(706)에서 위치된 테이블과 소스 테이블과 관련된 자신의 외부 키를 등록한다(단계(708)). 도 12에 도시된 예에서, 컴퓨터는 이동 경로를 완성하기 위해 고용인 테이블(540)에 대해 외부 키로서 관리자 테이블(550)의 사회_보장_번호를 등록할 것이다.

만일 단계(704)에서의 결과가 아니오이면, 컴퓨터 시스템은 단계(710)에서, 컴퓨터 시스템은 이동 경로가 일_대_다수인 지의 여부를 판정한다. 만일 그 결과가 예이면, 컴퓨터 시스템은 단계(712)에서 목적지 테이블이 새로운 것(즉, 현재 스키마에서 발견되지 않는)인지의 여부를 판정한다. 만일 그 결과가 예이면, 정의되어야 할 추가의 이동 경로가 더 이상 없다. 만일 목적지 테이블이 새로운 것이 아니면, 컴퓨터 시스템은 단계(714)에서 목적지 테이블의 테이블_Id를 매치시키는 테이블_Id를 갖는 자식 테이블을 위해 현재 스키마를 탐색한다. 컴퓨터는 그후 단계(716)에서 소스 테이블에 대해 자식 테이블 및 외부 키 관계를

이동 경로에 추가한다. 도 7에 도시된 일-대-다수 예에서, 이동 경로는 소스로서 학생 테이블(172)과, 목적지로서 전공 테이블(176)을 리스팅하고, 전공 테이블의 학생_Id가 학생 테이블(172)에 대해 외부 키라는 사실을 리스팅한다.

만일 이동유형이 일-대-일 또는 일-대-다수가 아니면, 단계(720)에서 컴퓨터는 유형이 다수-대-일인 지의 여부를 판정한다. 만일 그 결과가 예이면, 단계(722)에서 소스 테이블에 대한 부모 테이블이 이동 경로에 추가되고 소스 테이블과 부모 테이블간의 외부 키 관계에 추가된다. 다음에, 부모 테이블의 학생_Id가 목적지 테이블의 학생_Id를 매치시키는 지의 여부를 판정한다. 만일 그 결과가 예이면 이동 경로는 완료된다. 만일 그 결과가 아니오이면, 단계(726)에서 컴퓨터는 부모 테이블이 객체 모델의 객체를 표현하는 지의 여부를 판정한다. 만일 그 결과가 아니오이면, 컴퓨터는 단계(722)로 되돌아가고 부모 테이블의 부모 테이블을 이동 경로 및 외부 키 관계에 추가한다. 당업자에 의해 이해될 수 있는 바와 같이, 단계(722)에서 객체모델의 객체를 표현하는 테이블과 소스 테이블간의 각각의 테이블을 이동 경로에 추가한다. 만일 단계(726)에서의 결과가 예이면, 컴퓨터는 상기 단계(706 및 708)에서 설명한 바와 같이 일-대-일 이동 유형으로 진행한다(박스 728로 나타낸 바와 같이).

만일 단계(720)에서의 결과가 아니오이면, 컴퓨터는 이동유형이 다수-대-다수 인지의 여부를 판정한다. 만일 그 결과가 예이면, 컴퓨터는 단계(742)에서 소스 테이블의 부모 테이블과 외부 키를 이동 경로에 추가한다. 단계(744)에서, 컴퓨터 시스템은 목적지 테이블이 새로운 것(즉, 현재 스키마에서 발견되지 않는)인지의 여부를 판정한다. 만일 그 결과가 예이면, 이동 경로는 완료된다. 만일 목적지 테이블이 새로운 것이 아니면, 컴퓨터 시스템은 단계(746)에서, 목적지 테이블_Id를 매치시키는 테이블_Id를 갖는 자식 테이블을 위해 현재 스키마를 탐색한다. 컴퓨터는 그후 단계(748)에서 부모 테이블에 대해 외부 키 관계 및 목적지 테이블을 이동 경로에 추가한다.

만일 단계(740)에서의 결과가 아니오이면, 이동을 복사로 인식하며 완료될 추가의 이동경로는 더 이상 없다.

사용자로 하여금 동시에 기존 관계에 변경을 가할 수 있게 한다. '전화번호' 및 '이름'으로 레이블링 된 성분을 이동시킨 후 사무실 객체(500) 및 관리자 객체(520)를 삭제시키느 예가 도 12에 나타나 있다. 데이터를 이동시키기 위해 필요한 모든 정보는 현재 스키마에서 이용할 수 있고, 사용자는 이러한 모든 변경을 한번에 할 수 있다.

이러한 확장을 지지하기 위해, 이동경로 탐색은 도 14C에 예시된 바와 같이 수정될 것이 요구된다. 중요한 차이는 경로가 정의될 때 까지 이용 불가한 이동 경로 유형이다. 제안된 스키마에서 초기의 탐색은 임의의 추가 정보없이 행해져야 하며, 즉 모든 제안된 테이블은 소스 칼럼 ID를 매치시키는 ID를 갖는 칼럼에 대해 탐색되어야 한다. 일단 테이블이 발견되면, 매칭 테이블 ID와 함께 대응 테이블은 현재 스키마에서 탐색되어야 한다. 현재 스키마의 소스 테이블 및 목적지 테이블이 성공적으로 식별된 후, 이동 경로는 두 테이블을 연결하기 위해 정의될 것이 요구된다. 만일 목적지 테이블이 새로운 것이면, 그 부모 테이블도 탐색되어야 한다. 어떠한 데이터 이동도 생기지 않는 결과로 되는 두 테이블이 현재 스키마에서 전혀 관련이 없는 경우도 가능하다. 소스 테이블로부터 시작하여, 탐색은 그 자식 테이블(일-대-다수), 인척 테이블(다수-대-다수),부모 테이블(다수-대-일) 또는 관련 객체 테이블(일-대-일 또는 다수-대-일)을 통해 수행되어야 한다. 이동경로 유형은 따라서 상기한 바와 같이 경로 방향 및 소스 카디널리티에 의거하여 발견될 수 있다.

도 14C는 세만틱 객체 모델로 이동된 칼럼을 탐색하도록 확장하기 위해 본 발명에 의해 수행된 단계를 도시한다. 단계(814)에서 시작하여, 탐색은 이동되어야 할 칼럼의 ID를 매치시키는 ID를 갖는 칼럼에 대해 제안된 스키마에서 수행된다. 단계(816)에서 컴퓨터 시스템은 테이블이 발견되었는 지의 여부를 판정한다. 만일 그 결과가 아니오이면, 데이터는 이동되지 않는다. 만일 단계(816)에서의 결과가 예이면, 컴퓨터 시스템은 테이블이 새롭게 제안된 스키마에사 생성되었는 지의 여부를 판정한다. 만일 그 결과가 예이면, 처리는 단계(818)로 가고 여기서 컴퓨터 시스템은 단계(816)에서 발견된 테이블의 부모 테이블이 새로운 것인 지의 여부를 판정한다. 만일 그 결과가 예이면, 어떠한 이동경로도 수행되지 않는다.

만일 제안된 스키마의 부모 테이블이 새로운 것이 아니면, 컴퓨터 시스템은 단계(822)에서 위치될 테이블로서 부모 테이블을 정의한다. 컴퓨터 시스템은 단계(824)에서 매칭 ID를 갖는 테이블에 대한 현재의 스키마를 탐색한다. 단계(826)에서, 컴퓨터 시스템은 테이블이 발견되었는 지의 여부를 판정한다. 만일 그 결과가 아니오이면, 데이터 이동은 수행되지 않는다. 만일 단계(826)에서의 결과가 예이면, 현재 스키마에 테이블이 있는 것을 의미하는, 단계(840)에서 컴퓨터 시스템은 소스 테이블을 이동경로에 추가한다. 단계(842)에서 컴퓨터 시스템은 목적지 테이블에 대한 자식 테이블의 소스테이블의 리스트를 탐색한다. 단계(844)에서 컴퓨터 시스템은 목적지 테이블이 소스 테이블의 임의의 자식 테이블에서 발견되었는 지의 여부를 판정한다. 만일 그 결과가 아니오이면, 컴퓨터 시스템은 소스 테이블이 부모 테이블을 갖는 지의 여부를 판정하고 만일 그 결과가 예이면 임의의 인척 테이블을 탐색한다. 만일 소스 테이블이 부모 테이블을 갖지 않는다면, 처리는 단계(838)로 되돌아 간다.

단계(828)에서, 소스 테이블에 대한 외부 키 관계 및 부모 테이블이 이동 경로에 추가된다. 단계(830)에서, 컴퓨터는 부모 테이블의 테이블 리스트를 통해 인척 테이블을 탐색한다. 단계(832)에서, 컴퓨터 시스템은 목적지 테이블이 발견되었는 지의 여부를 판정한다. 만일 그 결과가 아니오이면, 컴퓨터 시스템은 소스 테이블의 부모 테이블을 통해 목적지 테이블을 탐색하는 루프를 시작한다. 인덱스, 현재 테이블은 소스 테이블로서 초기화되고 각각의 부모 테이블의 부모로 리셋트된다. 단계(834)에서, 컴퓨터 시스템은 현재 테이블의 부모 테이블이 목적지 테이블을 매치하는 ID를 갖는다. 만일 그 결과가 아니오이면, 컴퓨터 시스템은 단계(836)에서 부모 테이블이 객체 모델의 세만틱 객체와 연관되는 지의 여부를 판정한다. 만일 그 결과가 아니오이면, 단계(837)에서 현재 테이블에 대한 외부 키 관계 및 부모 테이블은 이동경로에 추가된다. 처리는 단계(834)로 복귀한다.

만일 단계(836)에서의 결과가 예이면, 컴퓨터 시스템은 관련 객체 테이블을 탐색한다. 단계(848)에서, 컴퓨터 시스템은 목적지 테이블이 임의의 관련 객체 테이블에서 발견되는 지의 여부를 판정한다. 만일 그 결과가 아니오이면, 이동 경로는 완료된다. 만일 단계(848)에서의 결과가 예이면, 컴퓨터 시스템은 현재

테이블에 대한 외부 키 및 목적지 테이블을 이동경로에 추가한다. 만일 목적지 테이블이 자식 테이블의 현재 테이블, 인척 테이블 또는 임의의 부모 테이블에서 발견되면, 컴퓨터 시스템은 단계(846)에서 현재 테이블에 대한 외부 키 및 목적지 테이블을 이동경로에 추가한다.

도 14B를 참조하면, 링크 유형 이동을 위한 이동경로를 계산하기 위해 컴퓨터 시스템에 의해 수행된 단계가 도시되어 있다. 단계(770)에서 시작하여, 컴퓨터는 '링크'와 등가의 DataType 멤버 변수를 갖는 각각의 DMForeignData 사례를 분석하는 루프를 시작한다. 컴퓨터는 먼저 이동이 일-대-일 유형인지의 여부를 판정한다(단계(772)). 만일 그 결과가 예이면, 컴퓨터는 단계(774)에서 소스 테이블을 이동경로에 추가한다. 그후 단계(776)에서 컴퓨터는 목적지 테이블_Id에서와 동일한 테이블_Id를 갖는 테이블에 대한 현재 스키마를 탐색한다. 단계(778)에서 목적지 테이블은 소스와 목적지 테이블간의 외부 키 관계를 따라 이동경로에 추가된다.

만일 이동유형이 일-대-일이 아니면, 단계(782)에서 컴퓨터는 이동이 다수-대-일인 지의 여부를 판정한다. 만일 단계(782)에서의 결과가 예이면, 단계(784)에서, 컴퓨터는 목적지 테이블_Id에서와 동일한 테이블_Id를 갖는 테이블에 대한 현재 스키마를 탐색하여 목적지 테이블을 이동경로에 추가한다. 단계(786)에서, 컴퓨터는 소스 테이블과 목적지 테이블간의 외부 키 관계 및 소스테이블을 이동경로에 추가한다. 단계(788)에서, 소스 테이블의 부모 테이블은 외부 키 정보를 따라 이동경로에 추가된다. 컴퓨터는 그후 단계(790)에서 부모 테이블이 객체 모델의 객체를 표현하는 지의 여부를 판정한다. 만일 그 결과가 예이면, 이동경로는 완료된다. 만일 그 결과가 아니오이면, 컴퓨터는 단계(788)로 복귀하고 부모 테이블의 부모 테이블을 외부 키 정보를 따라 이동경로에 추가한다.

만일 이동경로가 일-대-일 또는 다수-대-일이 아니면, 컴퓨터는 단계(792)에서 이동이 대체 유형인 지의 여부를 판정한다. 만일 그 결과가 예이면,단계(794)에서 컴퓨터는 소스 테이블을 이동경로에 추가한다. 다음에, 단계(798)에서 컴퓨터는 소스 테이블의 부모 테이블과 외부 키 정보를 이동경로에 추가하고 목적지 테이블_Id와 동일한 테이블_Id를 갖는 자식 테이블에 대한 현재 스키마를 탐색한다. 단계(804)에서, 컴퓨터는 자식 테이블이 발견되었는 지의 여부를 판정한다. 만일 그 결과가 예이면, 이동경로는 완료된다. 만일 그 결과가 아니오이면, 컴퓨터 시스템은 단계(798)에서 발견된 부모 테이블이 자신의 고유 부모 테이블을 갖는 지의 여부를 판정한다(단계(806)). 만일 그 결과가 예이면, 컴퓨터는 단계(774,776 및 778)에서 개시된 일-대-일 링크 유형 이동으로서 진행한다(박스 808로 표시된 바와 같이). 테이블이 객체를 나타내지 않으면, 컴퓨터는 단계(798)로 진행한다.

마지막으로, 이동 유형이 일-대-일, 다수-대-일 또는 대체이면, 컴퓨터는 이동이 갱신유형인 것으로 안다. 만일 그렇다면, 이동경로는 단계(810)에서 소스 테이블을 추가하고 단계(812)에서 소스 테이블에 대한 외부 키 관계 및 목적지 테이블을 추가함으로써 완료된다.

일단 이동경로가 계산되면 컴퓨터는 객체 모델에 행해진 변경을 반영하기 위해 관계형 데이터베이스를 수정하는 기초 데이터베이스 관리 시스템에의해 실행되는 것인 SQL문을 어셈블링한다.

SPTable, DMData 및 DMForeignData 부류의 사례를 생성하여 초기화한 후, 컴퓨터 시스템은 사용자에 의해 대응 객체 모델에 행해진 변경을 반영하기 위해 기존의 관계형 데이터베이스 스키마의 테이블을 수정한다. 도 15A-15D는 기존의 관계형 데이터베이스 스키마를 수정하기 위해 컴퓨터 시스템에 의해 수행된 단계를 나타낸다. 도 15A를 참조하면, 컴퓨터 시스템은 단계(900)에서 DMTable 부류의 각각의 사례를 분석하는 루프를 시작한다. 단계(902)에서, 컴퓨터 시스템은 각각의 사례에 대한 임시 테이블을 생성한다. 단계(902)에서, 컴퓨터 시스템은 데이터를 소스 테이블로부터 임시 테이블로 이동시킨다. 루프는 컴퓨터 시스템이 모든 DMTable 사례가 처리되었는 지의 여부를 판정하는 단계(906)에서 종료한다.만일 그렇지 않다면, 처리는 단계(900)로 복귀하고 다음 임시 테이블이 생성된다.

일단 임시 테이블이 생성되면, 컴퓨터 시스템은 SPTable 부류의 사례를 분석하는 일련의 루프를 시작한다. 상기한 바와 같이, SPTable 부류는 새로운 테이블 생성과 데이터베이스의 기존의 테이블에서 수행되어야 할 모든 변경에 대해 컴퓨터 시스템에 알린다. 단계(908)에서 컴퓨터 시스템은 사례가 도 11A에 도시되고 상기한 ChangeTableList에 위치된 순서로 각각의 SPTable 부류의 사례를 분석한다. 다음에 컴퓨터 시스템은 실행 유형이 '탈락'으로 설정되었는 지의 여부를 판정한다(단계(910)). 만일 그렇다면, 컴퓨터 시스템은 단계(912)에서 부류에 대한 탈락 테이블 기능을 호출하고 테이블은 데이터베이스로부터 제거된다. 단계(914)에서, 컴퓨터 시스템은 SPTable 부류의 모든 사례가 분석되었는 지의 여부를 판정한다. 만일 그렇지 않다면, 처리는 단계(908)로 간다.

단계(916)에서 시작하여(도 15B) 컴퓨터는 SPTable 부류의 각각의 사례를 분석하는 새로운 루프를 시작한다. 단계(918)에서, 컴퓨터 시스템은 사례의 indexOperationList를 판독한다. 컴퓨터는 단계(920)에서 엔트리는 탈락되어야 할 인덱스를 필요로 하는 지의 여부를 판정한다. 만일 단계(920)에서의 결과가 예이면, 컴퓨터는 단계(922)에서 탈락 인덱스 기능을 호출한다. 일단 전체 indexOperationList가 분석되면(단계(924)를 참조), 컴퓨터 시스템은 단계(928)에서 foreignkeyOperationList를 판독한다. 컴퓨터 시스템은 단계(930)에서, 리스트의 엔트리가 탈락되어야 할 테이블의 외부 키를 필요로 하는 지의 여부를 판정한다. 만일 그 결과가 예이면, 컴퓨터 시스템은 단계(932)에서 탈락 외부 키 멤버 기능을 호출한다. 컴퓨터는 단계(934)에서 foreignkeyOperationList의 모든 엔트리가 처리되었는 지의 여부를 판정한다. 만일 그 결과가 아니오이면 처리는 단계(928)로 복귀한다. indexOperationList 및 foreignkeyOperationList의 모든 엔트리가 처리된 후, 컴퓨터 시스템은 단계(926)에서 모든 SPTable 부류 사례가 처리되었는 지의 여부를 판정한다. 처리는 모든 SPTable 부류 사례가 분석될 때 까지 단계(916)로 복귀한다.

도 15C를 참조하면, 컴퓨터 시스템은 모든 SPTable 부류의 각각의 사례를 처리할 때 까지 또다른 루프를 시작한다. 단계(934)에서, 컴퓨터 시스템은 사례의 ExecutionType가 '생성'과 동일한 지의 여부를 판정한다. 만일 그 결과가 예이면, 컴퓨터 시스템은 단계(936)에서 생성 테이블 기능을 호출한다. 컴퓨터는 단계(938)에서 SPTable 부류의 모든 사례가 분석되었는 지의 여부를 판정한다. 만일 그 결과가 아니오이면 처리는 단계(932)로 복귀한다.

단계(938)를 따라 컴퓨터 시스템은 SPTable 부류의 각각의 사례를 분석하기 위해 또다른 루프를

시작한다.(박스 940을 참조) 단계(942)에서 컴퓨터는 사례의 ExecutionType가 '변경'인 지의 여부를 판정한다. 만일 그 결과가 예이면, 컴퓨터 시스템은 단계(944)에서 변경 테이블 기능을 호출한다. 단계(946)에서 SPTable 부류의 각각의 사례가 분석된 때를 판정한다.

도 15D를 참조하면, 컴퓨터 시스템은 단계(950)에서 각각의 SPTable 부류 사례를 또다시 분석하기 위해 루프를 시작한다. 컴퓨터 시스템은 단계(952)에서 foreignkeyOperationList를 판독한다. 그후 컴퓨터 시스템은 단계(952)에서 foreignkeyOperationList의 엔트리가 테이블에 추가되어야 할 외부 키를 필요로 하는 지의 여부를 판정하는 루프를 시작한다(단계(956)). 만일 그 결과가 예이면, 컴퓨터 시스템은 단계(958)에서 추가 외부 키 멤버 기능을 호출한다. 컴퓨터 시스템은 단계(960)에서 foreignkeyOperationList의 모든 엔트리가 분석되었는 지의 여부를 판정한다. 단계(960)에서 컴퓨터 시스템은 SPTable 부류의 모든 사례가 처리되었는 지의 여부를 판정한다. 만일 그 결과가 아니오이면, 처리는 단계(950)로 간다.

상기한 바와 같이, SPTable 부류가 분석된 후, 컴퓨터 시스템은 단계(964)에서 DMTable 부류의 각각의 사례를 또다시 처리하는 루프를 시작한다. 단계(966)에서 컴퓨터 시스템은 초기에 임시 테이블로부터 생성된 데이터를 목적지 테이블에 삽입한다. 단계(968)에서 DMTable 부류의 각각의 엔트리가 처리되었는 지의 여부를 판정하고 처리는 단계(970)에서 종료한다. 마지막으로 단계(970)에서 모든 임시 테이블이 탈락된다.

도 15A의 단계(904)에서 상기한 바와 같이, 컴퓨터는 데이터를 소스 테이블로부터 DMTable 부류 사례에 의해 정의된 임시 테이블로 이동시키도록 동작한다. 아래에서 본 발명이 달성되는 법을 상세히 설명한다.

도 15A-15D의 단계들을 실행하기 전에, 각각의 DMTable 사례에 대한 dataList의 각각의 DMData 또는 DMForeignData는 다음의 4 부류로 된 SQL명령어를 생성하기 위해 분석된다.

INSERT INTO 임시 테이블 이름(목적지 컬럼 이름)

SELECT 소스 컬럼 이름

FROM 목적지 테이블, 소스 테이블 이름

WHERE 결합 조건

일단 임시 테이블이 생성되면, 단일 SQL INSERT 명령이 새로운 목적지 테이블에 대한 임시 테이블을 채우기 위해 실행된다.

SQL 문의 INSERT 및 SELECT 절은 각각의 DMData 또는 DMForeignData 사례로부터 획득된다. FROM 및 WHERE 절은 각각의 DMData 또는 DMForeignData 사례에 대해 정의된 특정 이동 경로 및 이동 유형에 좌우되어 상이하게 정의된다. 데이터 이동 유형의 상이한 값 유형에 대한 변동은 아래와 같이 요약된다.

<u>값 유형 이동경로</u>

일_대_일

소스 테이블은 단순히 소스 테이블과 목적지 테이블간의 결합 조건으로 결합된다. 다음 예는'PK','C1' 및 'C2'로 레이블링 된 3개의 칼럼이 'dstTable', 'srcTable1' 및 'srcTable2'로 레이블링 된 테이블로부터 'tempTable'로 레이블링 된 테이블로 이동되는 법을 보여준다. 이 예에서, 테이블 srcTable1 로부터의 'FK1' 및 테이블 srcTable1 로부터의 'FK2' 로 레이블링 된 칼럼은 테이블 'dstTable'의 PK 칼럼에 대한 외부 키이다.

```
INSERT INTO   tempTable(PK,C1,C2)
SELECT        dstTable.PK, srcTable1.C1, srcTable2.C2
FROM          srcTable1, srcTable2, dstTable
WHERE         srcTable1.FK1=dstTable.PK
AND           srcTable2.FK2=dstTable.PK;
```

일_대_다수

일_대_다수 값 유형 데이터 이동인 경우에, 소스 테이블로부터 목적지 테이블로 데이터의 칼럼을 이동시키는 데 필요한 SQL 문은 더욱 정교하다. 컴퓨터가 목적지 테이블이 현재 스키마에 존재하지 않는 것으로 판정하면, 대용 키를 갖는 새로운 테이블을 생성하는 것이 필요하다. 그러기 위해, 임시 테이블이 생성된다. 제 1 임시 테이블(아래의 예에서 keyTable)은 대용 키를 위한 순차 값을 보유하는 하나의 칼럼을 갖는다. 제 1 임시 테이블을 생성하기 위해, 컴퓨터는 상기 문과 유사한 SQL문을 실행한다.

```
CREATE TABLE        keyTable(PK int);
```

순차 대용 키 값은 SQL 문을 사용하여 제 1 테이블에 반복적으로 추가된다.

```
INSERT INTO         keyTable(PK)
VALUES               (?);          여기서 '?'는 정수이다.
```

각각의 반복에 대해, 컴퓨터 시스템은 고유 정수 값(여기선 물음표로 표기됨)을 생성하여 이 정수를 대용 키로서 제 1 임시 테이블에 삽입한다. 컴퓨터 시스템은 엔트리의 수가 이동되어야 할 소스 테이블의 레코드의 수와 동일 할 때 까지 상기 INSERT 문을 실행한다.

현재 표준 SQL 문법은 행 번호순으로의 선택을 허용하지 않기 때문에, 제 2 임시 테이블(pkTable) 및 보기(view)가 이용된다. 제 2 임시 테이블은 생성되어 복사된 srcTable로부터 제 2 임시 테이블로 이동되어야 할 칼럼(예에서 C1)의 널 이 아닌 엔트리와 연관된 주 키 값을 다음 SQL문을 이용하여 복사한다.

```
CREATE TABLE        pkTable(PK 유형); 여기서 '유형'은 주 키의 데이터
유형이다

INSERT INTO         pkTable(PK)

SELECT               srcTable1.PK

FROM                 srcTable1

WHERE                srcTable.C1 은 널이 아니다;
```

그후 보기는 표준 SQL 함수 Min()를 이용하여 주 키 칼럼의 최소값을 발견하여 단일 주 키 값을 선택하기 위해 제 1 임시 테이블에 생성된다. 그러나, 데이터베이스 프로그래밍 당업자는 표준 SQL 함수 Min()도 사용될 수 있음을 이해할 것이다.

```
CREATE VIEW         minPKView(PK)

ON   SELECT         Min(PK)

FROM           pkTable;
```

일단 보기가 완료되면, 컴퓨터 시스템은 수정된 목적지 테이블을 위한 템플레이트로서 사용될 제 3 임시 테이블을 생성한다.

제 3 임시 테이블이 표명된 후, 컴퓨터 시스템은 이동되어야 할 데이터를 포함하는 소스 테이블의 칼럼 C1 으로부터의 엔트리와, 제 3 임시 테이블의 주 키를 형성하는 대용 키 및 외부 키로서 사용되는 소스 테이블의 주 키를 삽입하는 루프를 시작한다. 다음은 이들 단계를 완성하는 SQL 문의 예이다. WHERE의 '?'는 실행되는 문으로서 정수번호로 대체된다. 그 후 값은 다음 삽입을 위해 증가된다.

```
INSERT INTO         tempTable(C1,FK,PK)

SELECT               srcTable1.C1, srcTable.PK, keyTable.PK

FROM                 srcTable1, keyTable2, minPKView

WHERE                keyTable.key=?

AND                  srcTable1.PK=minPKView.PK;
```

각각의 행을 삽입한 후, 'pkTable'의 대응 행은 삭제된다. 이 프로세스는 pkTable이 엠프티가 될 때까지 반족된다. 모든 임시 테이블 및 보기는 데이터베이스로부터 삭제된다.

만일 목적지 테이블이 이미 현재 데이터베이스 스키마에 존재한다면, SQL문은 소스 테이블 ' srcTable1' 로 부터의 칼럼 C1을 간단히 'tempTable'인 임시 테이블로 이동시킬 것이 요구된다.

```
INSERT INTO         tempTable(FK,PK,C1)

SELECT               dstTable.FK, dstTable.PK, srcTable.C1

FROM                 dstTable, srcTable1

WHERE                srcTable1.PK=dstTable.FK
```

테이블 'tempTable'이 셋업된 후, 전체 테이블은 목적지 테이블로 복사되고 임시 테이블은 탈락될 수 있다.

다수_대_일

다수_대_일 값 이동은 다중값을 갖는 경우로부터 단일 값을 갖는 경우로 성분의 카디널리티 변경결과로서 생긴다. 이 변경을 데이터베이스에 반영하기 위해, 자주 데이터를 삭제할 것이 필요하다. 질의는 어느 데이터 값 소스 테이블로부터 목적지 테이블로 복사되고 어느 데이터 값이 버려질 것인가가 된다. 결과적으로, 본 발명의 실시예는 언제나 저장되어야 할 레코드로서 최소 주 키를 갖는 레코드를 선택한다.

데이터 이동을 성취하기 위해, 소스 테이블의 주키의 데이터베이스 보기가 생성된다. 이동경로는 다수 테이블을 포함할 수 있기 때문에, 소스 테이블과 목적지 테이블간의 외부 키 관계를 지정하는 것이 필요하다. 따라서, 보기는 다음 형태와 같은 SQL 문을 이용하여 전체 이동경로를 포함하도록 구성된다.

```
CREATE VIEW         pkView(FK,minPK)

ON   SELECT         T1.FK,Min(srcTable.PK)

FROM           T1,T2,..., Tn, srcTable

WHERE          T1.PK=T2.FK

.....

AND              Tn.PK = srcTable.FK

GROUP BY         T2.FK;
```

여기서T1...Tn은 목적지 테이블을 제외한 이동경로의 테이블이다.

보기가 정의된 후, 컴퓨터 시스템은 선택된 레코드를 목적지 테이블에 추가하기 위해 INSERT SQL명령어를 실행한다. 이 보기를 이용하여, INSERT 명령어는 다음과 같을 것이다.

```
INSERT INTO         tempTable(PK,C1,C2)
SELECT               dstTable.PK, srcTable.C1, dstTable.c2
FROM                 srcTable, dstTable,pkView
WHERE                srcTable1.PK=pkView.minPK
AND                  dstTable.PK=pkView.FK
```

상기 SQL 문을 실행한 후, 'pkView'는 삽입 동작을 완료하기 위해 탈락될 것이 필요하다.

다수_대_다수

다수_대_다수 유형 데이터 이동은 사실상 두 데이터 셋트를 병합하는 것이다. 그러므로, 동일 방식으로 소스 및 목적지 테이블로 부터의 레코드를 연관시키는 것이 필요하다. 결과적으로, 본 발명은 다음과 같이 부모 사례당 단순한카티시안 곱을 사용한다.

```
INSERT INTO         tempTable(FK,PK,C1,C2,PK2)
SELECT               srcTable.FK, srcTable.PK, srcTable.C1,
srcTable.C2, dstTable.PK
FROM                 srcTable, dstTable
WHERE                srcTable1.FK=dstTable.FK
```

병합된 테이블은 이중 주 키 값결과로 되는 카티시안 곱으로 인해 생성된 새로운 레코드를 갖는다. 이중 주 키 값은 새로운 고유 값으로 제할당될 것이 요구된다.

이 목적을 달성하기 위해, 각각의 소스 테이블로 부터의 주 키도 임시 테이블에 복사된다. 두 개(또는 그 이상)의 주 키 칼럼은 목적지 테이블의 대용 키 값을 선택적으로 갱신하기 위해 고유 상태로 존재한다. 컴퓨터는 먼저 다음 문을 사용하여 임시 테이블을 생성한다.

```
CREATE TABLE        pkTable(FK int, PK int, PK2 유형)
```

여기서 '유형'은 제 2 주 키의 데이터유형이다. 다음 주키는 다음 문을 사용하여 임시 테이블에 삽입된다.

```
INSERT INTO         pkTable(FK,PK,PK2)
SELECT               FK,PK,PK2
FROM                 tempTable;
```

두 개(또는 연관된 테이블의 수에 좌우되는 그 이상의)보기는 순차적으로 생성된다. 제 1 보기는 각각의 외부 키에 대해 즉, 부모 레코드 당 제 1 주 키의 최소 값을 선택한다.

```
CREATE VIEW         minPKView(FK,minPK)
ON   SELECT         FK,Min(PK)
FROM           pkTable
GROUP BY        FK;
```

제 2 보기는 제 1 주키의 각각의 최소값에 대해 제 2 주키의 최소값을 선택한다.

```
CREATE VIEW         minPKView(minPK, minPK2)
ON   SELECT         minPK, Min(PKTable.PK2)
FROM           minPKView, PKTable
GROUP BY        minPK;
```

이 갱신은 주 키 테이블이 엠프티로 될 때 까지 각각의 실행 후에 정수인 '?'를 갖는 다음과 같은 두 개의 SQL 명령을 반복적으로 실행하므로써 수행된다. 이 키는 기존의 주키 값 더하기 일의 최대값으로 초기화 된다.

```
UPDATE        tempTable
SET           PK = ?
WHERE         1<(SELECT    COUNT(*)
FROM minPKView2
WHERE minPKView2.minPK = tempTable.PK
AND   minPKView2.minPK2 =
tempTable.PK2)
DELETE FROM  pkTable
```

```
1<(SELECT    COUNT(*)
FROM minPKView2
WHERE minPKView2.minPK = pkTable.PK
AND   minPKView2.minPK2 = pkTable.PK2;
```

링크 유형 이동

외부 키가 다른 객체 테이블로 이동될 때, SQL 삽입 명령어는 상수로서 소스 외부 키 데이터를 이용하여 소스 테이블로부터 주 키값을 복사하기 위해 상이하게 정의될 필요가 있다. 이것은 각각의 DMForeignData 사례의 이동경로에 좌우된다. 다음은 각각의 링크 유형 이동이 본 발명에 의해 취급되는 것을 나타낸다.

일_대_일

일_대_일 이동의 경우에, 소스 테이블 주키 값은 새로운 외부 키 값으로서 삽입된다. 다음의 SQL 문은 테이블 'srcTable'의 'FK'로 레이블링 된 칼럼에서 발견된 외부 키 값이 테이블 'dstTable'의 'PK'로 레이블링 된 칼럼으로 이동된다.

```
INSERT INTO         tempTable(PK,FK)
SELECT               dstTable.PK, srcTable.PK,
FROM                 dstTable, srcTable,
WHERE                dstTable.PK = srcTable.FK
```

다수_대_일

상기한 바와 같이, 다수_대_일 이동은 흔히 데이터베이스의 데이터가 삭제될 것이 요구된다. 그러므로, 보기는 다음 형태와 같은 SQL 문을 사용하여 기타 객체 사례당 하나로 다수의 레코드를 감소시키도록 생성된다.

```
CREATE VIEW         minPKView(dstFK, minPK)
ON   SELECT         dstFK, Min(srcObjTable.PK)
FROM           srcObjTable, ..., Tn, srcTable
WHERE          srcObjTable.PK = T1.FK
AND            ...
AND            TN.PK = srcTable.Fk
GROUP BY        dstFK;
```

상기 보기를 사용하여, 주 INSERT 명령어는 다음과 같이 된다.

```
INSERT INTO         tempTable(PK,FK)
SELECT               dstTable.PK, srcTable.PK
FROM                 dstTable, srcTable, minPKView
WHERE                srcTable.PK = minPKView.minPK
AND                  dstTable.PK = minPKView.dstFK;
```

갱신

이 이동 유형은 참조된 테이블에서 발견된 주 키에서의 몇몇 변경으로 인해 기존의 외부 키가 새로운 값을 필요로 할 때 이용된다. 새로운 키 값은 참조된 테이블에서 PK가 주 키값을 보유하는 칼럼이고 FK가 외부 키 칼럼의 이름인 경우에 아래에 나타난 형태를 갖는 SQL 문을 사용하는 기존의 키 제약조건을 이용하여 단순히 복사된다.

```
INSERT INTO         tempTable(FK)
SELECT               refTable.PK
FROM                 srcTable, refTable
WHERE                srcTable.FK = refTable.PK;
```

대체

이 이동유형은 부모 테이블의 변경 즉, 성분 테이블이 다른 부모 테이블로 이동되는 것으로 인해 외부 키가 새로운 정의로 대체될 때 이용된다. 새로운 키 값은 아래에 나타난 바와 같은 SQL 문을 실행하여 기존의 키 제약조건을 이요하여 복사되고, 여기서 T1 ... TN은 상기와 같이 계산된 이동경로에서 발견된 테이블이다.

```
INSERT INTO         tempTable(FK)
SELECT               refTable.PK
```

```
FROM            srcTable, T1, ..., Tn, refTable
WHERE           srcTable.FK = T1.PK
AND             ...
AND             T1.FK = refTable.PK
```

현재 데이터베이스가 대응 객체 모델에 행해진 변경을 반영하기 위해 수정된 후, 제안된 스키마는 추가 객체 모델 변경에 대한 비교로서 사용되기 위해 현재 스키마로서 저장된다. 상기 설며으로부터 알 수 있는 바와 같이, 본 발명은 대응 객체 모델에 행해진 변경에 기초하여 기존의 데이터베이스 스키마를 수정하도록 동작한다. 본 발명은 사용자로 하여금 테이블, 외부 키,대용 키,등과 같은 일반적인 관계형 데이터베이스 개념을 이해할 필요없이 데이터베이스를 용이하게 갱신할 수 있게 한다. 본 발명은 관계형 데이터베이스에 대한 경험이 없는 사용자도 용이하게 관계형 데이터베이스를 사용할 수 있게 하는 것으로 여겨진다. 다.

본 발명의 바람직한 실시예가 예시되고 설명되었을 지라도, 본 발명의 범위 및 정신으로부터 벗어나지 않고 다양한 변경이 이루어질 수 있음이 인식되어야 한다.

(57) 청구의 범위

청구항 1

대응 객체모델에 행해진 변경을 반영하기 위해 기존의 관계형 데이터베이스를 수정하는 방법에 있어서,

관계형 데이터베이스에 저장된 데이터에 대한 항목의 카테고리를 표현하며 항목에 대해 관계형 데이터베이스에 저장된 데이터를 정의하는 적어도 하나의 객체를 포함하는 객체모델을 컴퓨터 시스템의 메모리에 저장하는 단계;

객체모델의 영상 표현을 디스플레이하는 단계;

하나 이상의 관계형 테이블에 포함된 하나 이상의 컬럼과 기존의 관계형 데이터베이스에 포함된하나 이상의 관계형 테이블을 정의하는 현재 관계형 데이터베이스 스키마를 컴퓨터 시스템의 메모리에 저장하는 단계;

사용자에 의해 행해진 객체모델에 대한 수정을 검출하여 수정된 객체모델 에 대응하는 제안된 관계형 데이터베이스 스키마를 자동적으로 생성하고, 현재의 관계형 데이터베이스 스키마와 제안된 관계형 데이터베이스 스키마를 자동적으로 비교하는 단계;와

객체모델에 대한 수정이외에 사용자로부터 어떠한 추가 입력 없이 제안된 관계형 데이터베이스 스키마와 현재 관계형 데이터베이스 스키마의 비교에 기초하여 관계형 데이터베이스를 자동적으로 수정하는 단계를 포함하는 것을 특징으로 하는 방법.

청구항 2

제 1항에 있어서, 제안된 관계형 데이터베이스 스키마와 현재 관계형 데이터베이스 스키마는 관계형 데이터베이스 테이블의 리스트를 포함하고, 제안된 관계형 데이터베이스 스키마와 현재 관계형 데이터베이스 스키마를 비교하는 단계는,

현재 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블에 있는 리스트의 각각의 엔트리가 제안된 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블에 있는 리스트에서 발견되는 지의 여부를 자동적으로 판정하는 단계;와

현재 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블의 리스트에는 있지만 제안된 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블의 리스트에는 없는 각각의 테이블을 관계형 데이터베이스로부터 자동적으로 삭제하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 3

제 2항에 있어서, 현재 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블의 리스트에서 발견되지 않는 제안된 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블의 리스트에 임의의 엔트리가 있는 지의 여부를 자동적으로 판정하는 단계;와

제안된 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블의 리스트에는 있지만 현재 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블의 리스트에는 없는 각각의 엔트리에 대해 관계형 데이터베이스 테이블을 관계형 데이터베이스에 자동적으로 추가하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 4

제 3항에 있어서, 현재 관계형 데이터베이스 스키마 및 제안된 관계형 데이터베이스 스키마는 각각의 테이블에 포함된 각각의 칼럼의 리스트를 포함하고,현재 관계형 데이터베이스 스키마와 제안된 관계형 데이터베이스 스키마를 비교하는 단계는,

현재 스키마내의 테이블에 포함된 칼럼의 리스트의 각각의 엔트리가 현재 스키마내의 테이블에 포함된 칼럼의 리스트에서 발견되는 지의 여부를 자동적으로 판정하는 단계;와

현재 데이터베이스 스키마내의 테이블에는 포함되었지만 제안된 데이터베이스 스키마내의 테이블에는 포함되지 않은 칼럼을 관계형 데이터베이스로부터 자동적으로 탈락시키는 단계를 더 포함하는 것을 특징으

로 하는 방법.

청구항 5

제 4항에 있어서, 컴퓨터 시스템은 관계형 데이터베이스로부터 탈락되어야 할 칼럼인 지의 여부를 판정하고, 상기 방법은

관계형 데이터베이스로부터 탈락되어야 할 칼럼이 데이터를 포함하는 지의 여부를 자동적으로 판정하는 단계로서, 만일 이 단계에서의 결과가 예이면 탈락되어야할 칼럼이 제안된 관계형 데이터베이스 스키마의 다른 테이블로 이동되었는 지의 여부를 판정하기 위해 제안된 관계형 데이터베이스 스키마의 테이블의 리스트를 탐색하며;

탈락되어야 할 칼럼이 제안된 관계형 데이터베이스 스키마의 다른 테이블에서 발견되었다면 데이터를 관계형 데이터베이스의 소스 테이블로부터 관계형 데이터베이스의 목적지 테이블로 자동적으로 이동시키는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 6

제 5항에 있어서, 데이터를 상기 소스 테이블로부터 상기 목적지 테이블로 자동적으로 이동시키는 단계는,

관계형 데이터베이스에 임시 테이블을 자동적으로 생성하는 단계;

상기 임시 테이블에 데이터를 자동적으로 이동시키는 단계;

상기 목적지 테이블이 저장되어야 할 임의의 데이터를 포함하는 지의 여부를 자동적으로 판정하는 단계로서, 만일 이 단계에서의 결과가 예이면 저장되어야 할 임의의 데이터를 임시 테이블에 이동시키는 단계;

관계형 데이터베이스로부터 목적지 테이블을 자동적으로 탈락시키는 단계;

소스 테이블로 부터의 데이터 및 저장되었어야 할 데이터를 보유하기 위해 칼럼을 포함하는 새로운 목적지 테이블을 자동적으로 생성하는 단계;

데이터베이스를 임시 테이블로부터 새로운 목적지 테이블로 자동적으로 이동시키는 단계; 및

관계형 데이터베이스로부터 임시 테이블을 자동적으로 탈락시키는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 7

제 1항에 있어서, 관계형 데이터베이스는 관계형 데이터베이스의 다른 테이블에 대해 외부 키를 갖는 하나 이상의 테이블을 포함하며, 상기 방법은;

수정되어야 할 각각의 테이블을 포함하는 리스트를 컴퓨터의 메모리에 자동적으로 생성하는 단계를 더 포함하고, 여기서 리스트의 엔트리는 외부 키가 테이블을 참조한 후 데이터베이스의 다른 테이블에 대한 외부 키를 포함하는 테이블이 수정되도록 배열되는 것을 특징으로 하는 방법.

청구항 8

제 5항에 있어서,

데이터를 소스 테이블로부터 목적지 테이블로 이동시키는 단계는,

소스 테이블과 목적지 테이블간의 외부 키 관계를 지시하는 소스 테이블과 목적지 테이블간의 이동 경로를 자동적으로 판정하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 9

각각이 관계형 데이터베이스에 저장된 정보를 표현하며, 관계형 데이터베이스에 저장된 정보를 표현하는 적어도 하나의 성분을 갖는 적어도 하나의 객체를 포함하는 대응 객체모델에 행해진 변경을 반영하기 위해 현재 관계형 데이터베이스 스키마를 수정하기 위한 시스템에 있어서,

처리 유닛;

이 처리 유닛에 결합된 메모리;

현재 관계형 데이터베이스 스키마에 대응하는 객체모델의 표현을 산출하기 위해 중앙처리유닛에 의해 구동되는 디스플레이;

사용자로 하여금 객체모델을 수정할 수 있게하는 데이터 엔트리 수단;

중앙처리유닛으로 하여금 수정된 객체모델을 분석하기 위해 그리고 제안된 관계형 데이터베이스 스키마를 자동적으로 생성하게 하며, 또한 중앙처리유닛으로 하여금 제안된 관계형 데이터베이스 스키마와 현재 관계형 데이터베이스 스키마를 자동적으로 비교하게하며 객체모델에 대한 수정이외의 추가의 사용자 입력없이 수정된 객체모델에 대응하도록 현재 관계형 데이터베이스 스키마를 자동적으로 수정하게 하는 메모리에 배치된 프로그램된 명령 셋트를 포함하는 것을 특징으로 하는 시스템.

청구항 10

제 9항에 있어서, 현재 관계형 데이터베이스 스키마는 각각이 데이터가 저장될 적어도 하나의 칼럼을 갖는 복수의 테이블을 포함하고, 여기서 상기 명령 셋트는 컴퓨터로 하여금 테이블이 수정된 객체모델에 대응하도록 수정되어야 할 현재 관계형 데이터베이스 스키마를 포함하는 순서를 지시하는 메모리내의 리스

트를 생성하게 하며, 여기서 상기 리스트는 현재 관계형 데이터베이스 스키마의 다른 테이블에 대해 외부 키를 갖는 테이블이 외부 키가 참조하는 테이블 다음에 수정되도록 배열되는 것을 특징으로 하는 시스템.

청구항 11

제 9항에 있어서, 제안된 관계형 데이터베이스 스키마 및 현재 관계형 데이터베이스 스키마는 관계형 데이터베이스 테이블을 포함하고, 명령 셋트는 ,

현재 관계형 데이터베이스 스키마에 포함된 임의의 관계형 데이터베이스 테이블이 제안된 관계형 데이터베이스 스키마에 있는 지의 여부를 판정하기 위해제 및 현재 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블의 리스트를 탐색하고,

현재 관계형 데이터베이스 스키마에는 있지만 제안된 관계형 데이터베이스 스키마에는 없는 임의의 관계형 데이터베이스 테이블을 현재 관계형 데이터베이스 스키마로부터 탈락시키는 SQL 문을 생성하므로써, 컴퓨터 시스템으로 하여금 제안된 관계형 데이터베이스 스키마와 현재 관계형 데이터베이스 스키마를 비교하게 하는 것을 특징으로 하는 시스템.

청구항 12

제 11항에 있어서, 명령 셋트는,

현재 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블의 리스트에서 발견되지 않는 제안된 관계형 데이터베이스 스키마의의 임의의 관계형 데이터베이스 테이블이 있는 지의 여부를 판정하기 제안된 및 현재 관계형 데이터베이스 테이블의 리스트를 탐색하고,

현재 관계형 데이터베이스 스키마에는 없지만 제안된 관계형 데이터베이스 스키마에 있는 각각의 테이블을 위해 관계형 데이터베이스 테이블을 현재 관계형 데이터베이스 스키마에 추가하는 SQL 문을 생성하게 하므로써, 컴퓨터 시스템으로 하여금 제안된 관계형 데이터베이스 스키마와 현재 관계형 데이터베이스 스키마를 비교하게 하는 것을 특징으로 하는 시스템.

청구항 13

제 11항에 있어서, 현재 관계형 데이터베이스 스키마와 제안된 관계형 데이터베이스 스키마는 각각의 관계형 데이터베이스 테이블에 포함된 각각의 칼럼의 리스트를 포함하고, 명령 셋트는

제안된 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블에서 발견되지 않는 현재 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블에 임의의 칼럼이 포함되었는 지의 여부를 판정하기 위해 현재 관계형 데이터베이스 스키마와 제안된 관계형 데이터베이스 스키마의 각각의 테이블에 대한 리스트를 탐색하고,

제안된 관계형 데이터베이스 스키마의 테이블에는 없지만 현재 관계형 데이터베이스 스키마의 테이블에 포함된 칼럼들을 현재 관계형 데이터베이스 스키마로부터 탈락시키는 SQL 문을 생성하므로써, 컴퓨터 시스템으로 하여금 제안된 관계형 데이터베이스 스키마와 현재 관계형 데이터베이스 스키마를 비교하게 하는 것을 특징으로 하는 시스템.

청구항 14

제 4항에 있어서, 명령 셋트는,

칼럼이 데이터를 포함하는 데이터베이스로부터 탈락되어야 하는 지의 여부를 판정하고, 만일 그렇다면 칼럼이 제안된 관계형 데이터베이스 스키마의 다른 테이블로 이동되었는 지의 여부를 판정하기 위해 제안된 관계형 데이터베이스 스키마의 관계형 데이터베이스 테이블의 몇몇을 탐색하고,

탈락되어야 할 칼럼이 제안된 관계형 데이터베이스 스키마에 정의된 컴퓨터 시스템에서 발견된다면 컴퓨터 시스템에 의해 동작될 때 관계형 데이터베이스의 소스 테이블로부터 관계형 데이터베이스의 목적지 테이블로 데이터를 이동시키는 SQL 문을 발생시키므로써, 컴퓨터 시스템이 동작되게 하는 것을 특징으로 하는 방법.

청구항 15

제 14항에 있어서, 명령 셋트는,

관계형 데이터베이스의 임시 테이블을 생성하고;

데이터베이스를 임시 테이블로 이동시키고;

목적지 테이블이 저장되어야 할 임의의 데이터를 포함는 지의 여부를 판정하여 만일 포함한다면 저장될 데이터를 임시 테이블로 이동시키고;

관계형 데이터베이스로부터 목적지 테이블을 탈락시키고;

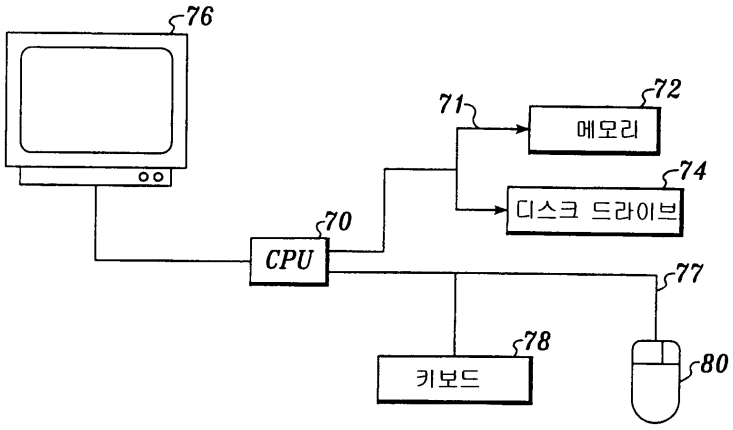
소스 테이블로 부터의 데이터 및 저장되었어야 할 데이터를 보유하기 위해컬럼을 포함하는 새로운 목적지 테이블을 생성하고;

데이터베이스를 임시 테이블로부터 새로운 목적지 테이블로 이동시키고;그리고

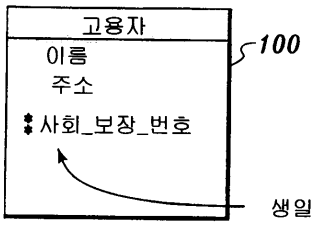
관계형 데이터베이스로부터 임시 테이블을 탈락시킬, 일련의 SQL문을 컴퓨터 시스템으로 하여금 생성하게 하는 것을 특징으로 하는 시스템.

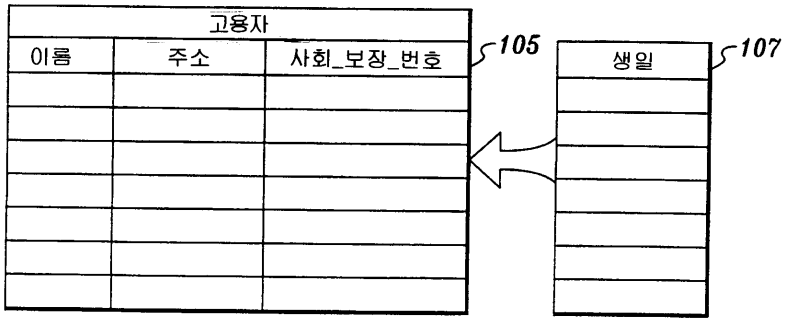
도면

도면1



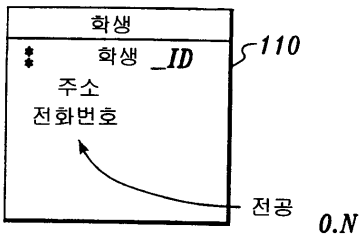
도면2





주 키 = 사회_보장_번호

도면3



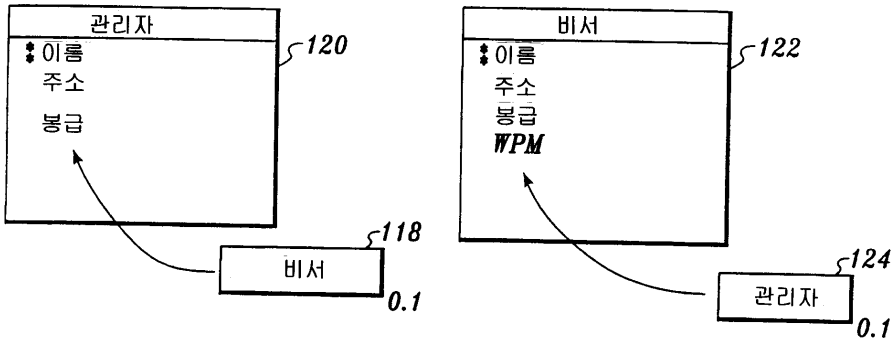
학생			
학생 _ID	주소	전화번호	112

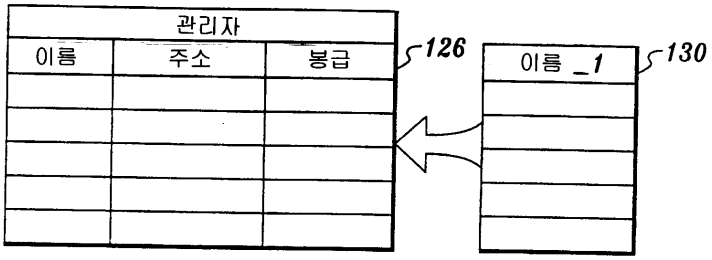
주 키 = 학생_ID

전공			
대용 키	전공	학생_ID	114

주 키 = 대용 키
학생_ID는 학생의 학생_ID에 대한 외부키이다

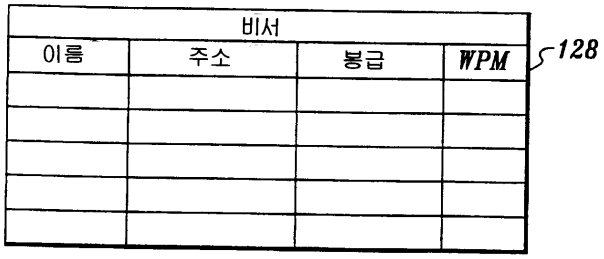
도면4





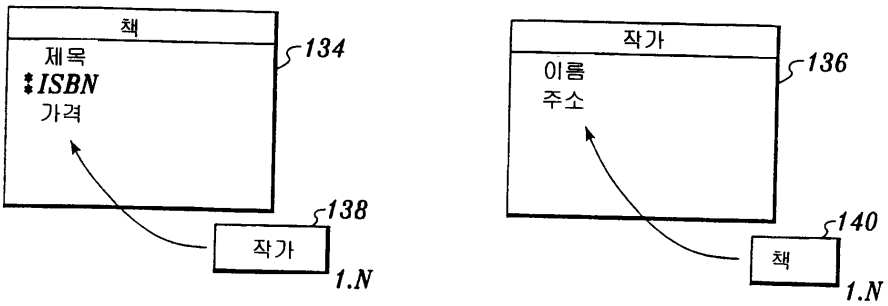
주 키 = 이름

이름_1은 비서의 이름에 대한 외부키이다.



주 키 = 이름

도면5



책			144
제목	ISBN	가격	

주 키 = ISBN

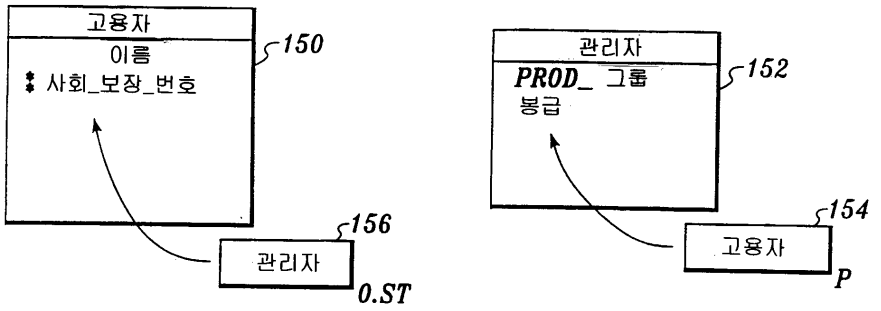
작가			146
대용 키	이름	주소	

주 키 = 대용 키

작가 - 책		148
대용 키	ISBN	

주키 = (대용 키)
대용키는 작가의 대용 키에 대한 외부키이다.
ISBN은 책의 ISBN에 대한 외부키이다.

도면6



고용자		158
이름	사회_보장_번호	

주 키 = 사회_보장_번호

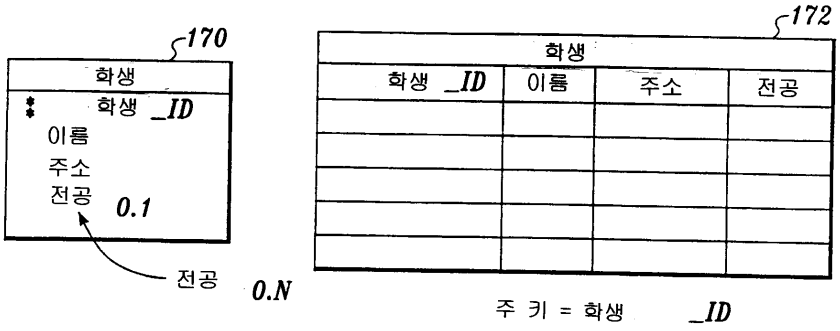
관리자			160
대용 키	PROD_ 그룹	봉급	

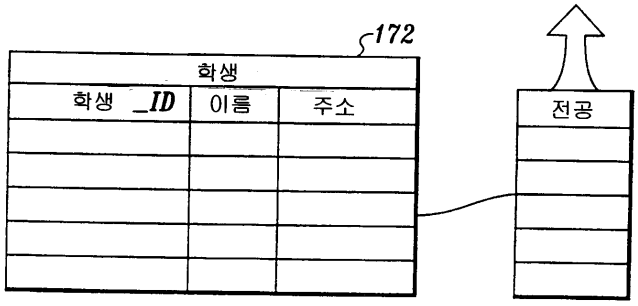
사회_보장-번호	162

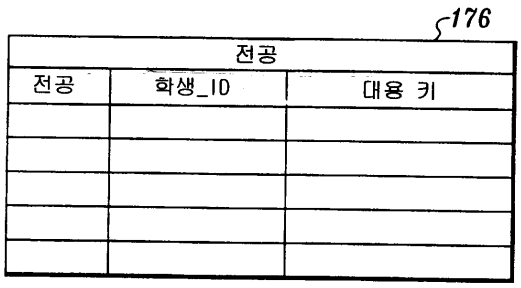
주 키 = 대용 키

사회_보장_번호는 고용자의 사회_보장_번호에 대한 외부키이다.

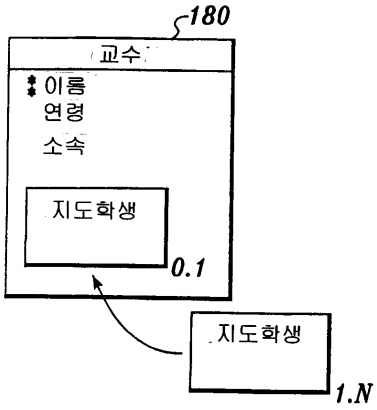
도면7

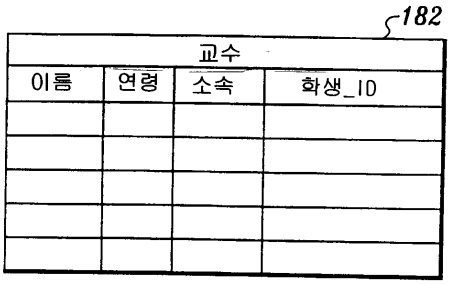

170
학생
학생 _ID
이름
주소
전공
0.1
전공
0.N
172
학생
학생 _ID
이름
주소
전공
주 키 = 학생 _ID


172
학생
학생 _ID
이름
주소
전공
주 키 = 학생_ID

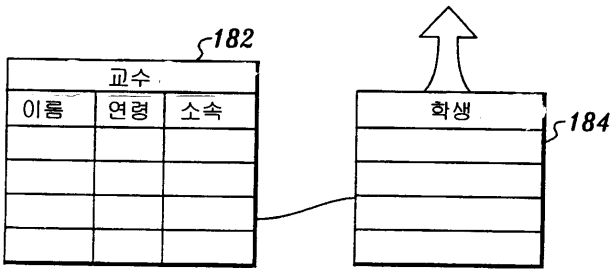

176
전공
전공
학생_ID
대용 키
주 키 = 대용 키
학생_ID는 학생의 학생_ID에 대한 외부키이다.

도면8

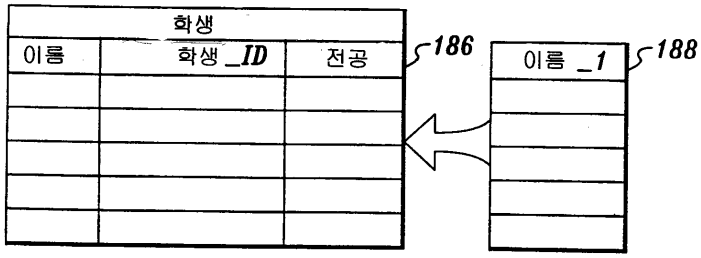




주 키 = 이름
학생_ID는 학생의 학생_ID에 대한 외부키이다.

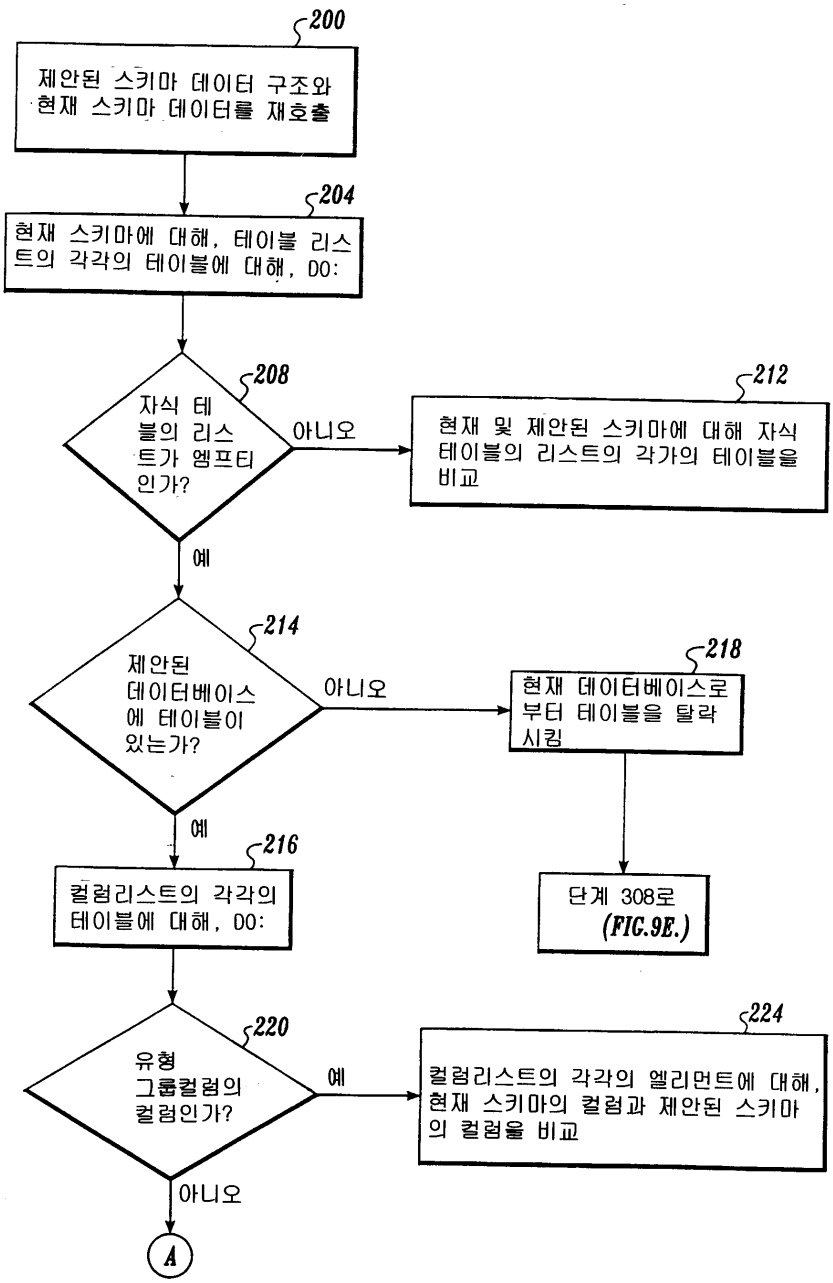


주키 = 이름
학생_ID는 학생의 학생_ID에 대한 외부키이다.

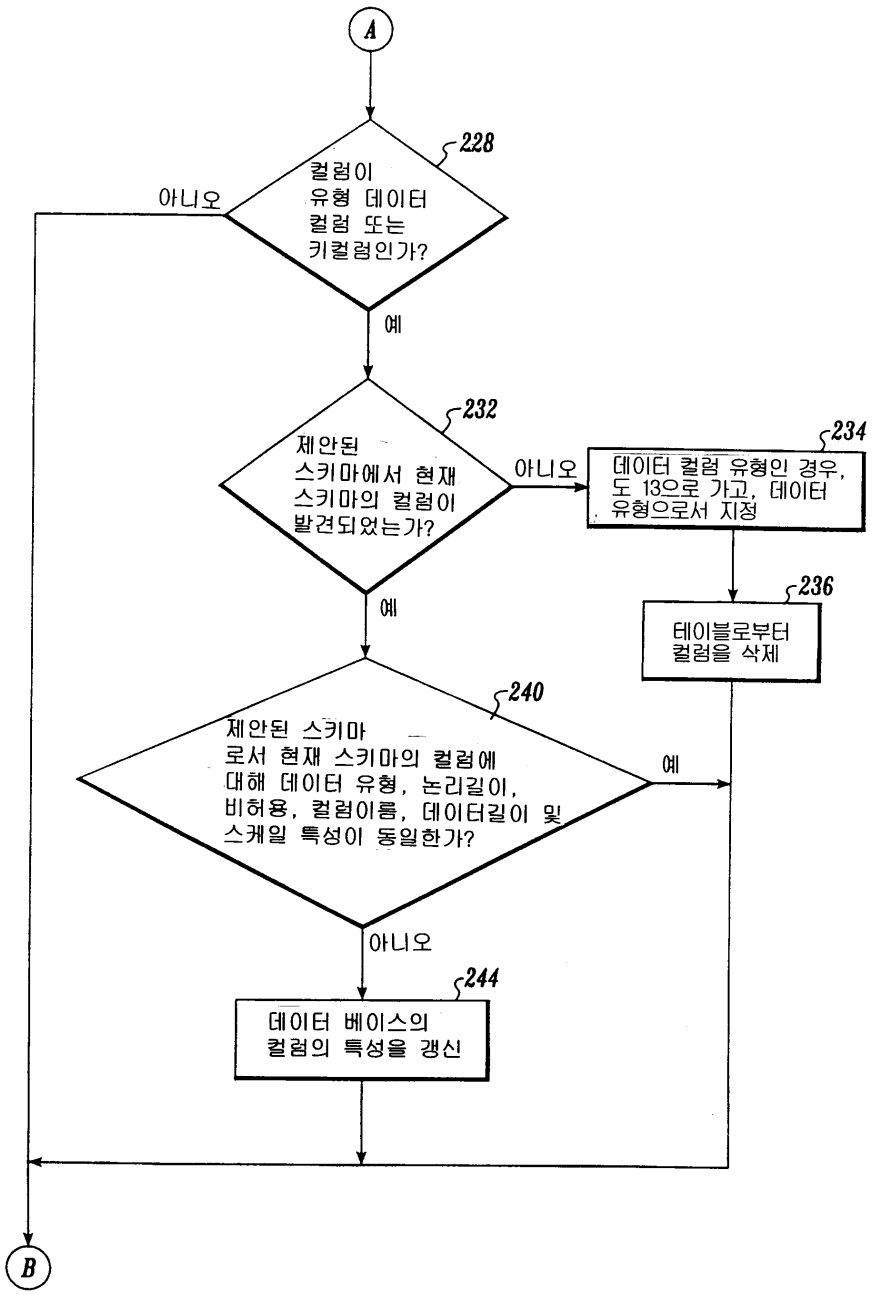


주키 = 학생_ID
이름_1은 교수의 이름에 대한 외부키이다.

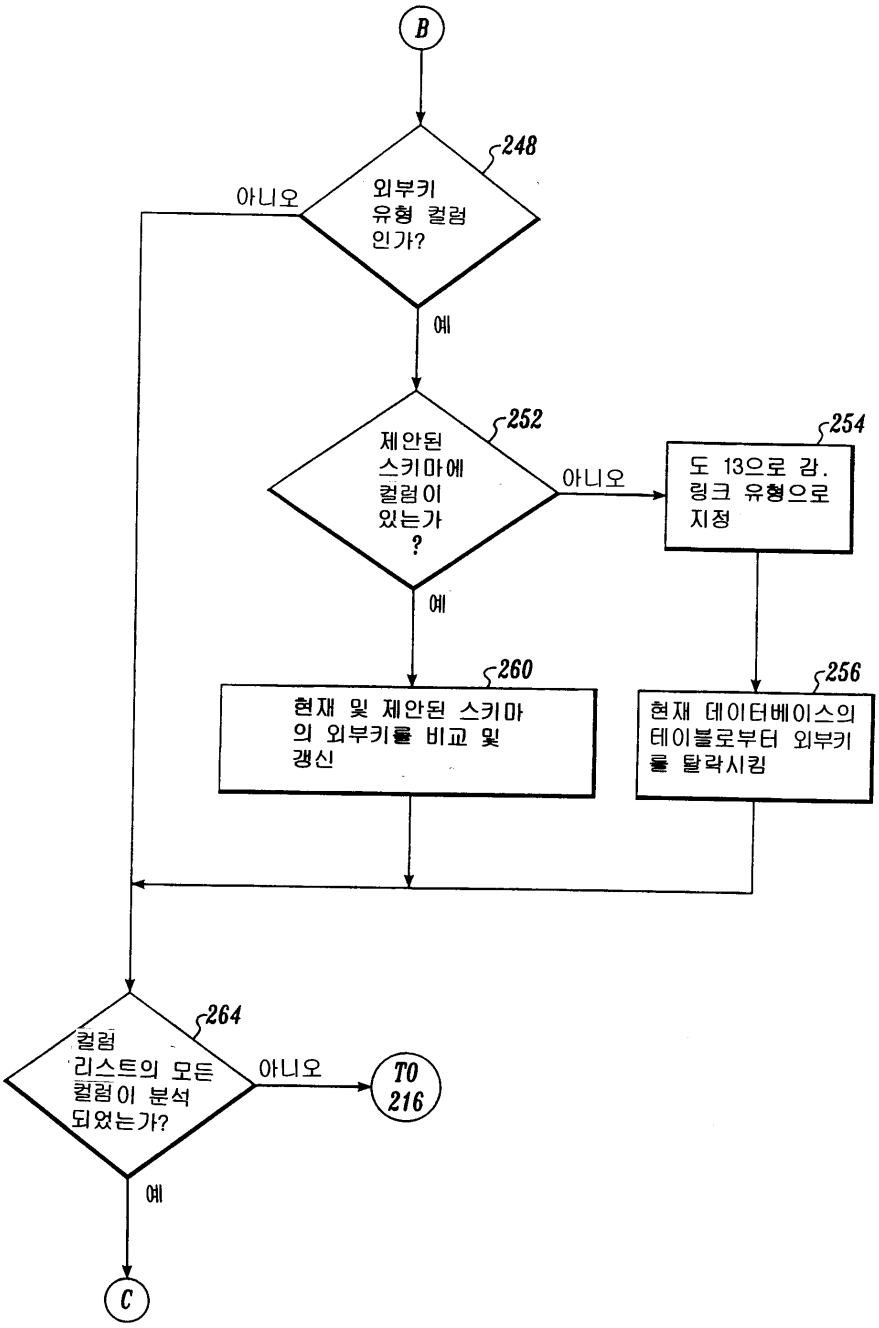
도면9A


200
제안된 스키마 데이터 구조와 현재 스키마 데이터를 재호출
204
현재 스키마에 대해, 테이블 리스트의 각각의 테이블에 대해, DO:
208
자식 테이블의 리스트가 엠프티인가?
아니오
212
현재 및 제안된 스키마에 대해 자식 테이블의 리스트의 각가의 테이블을 비교
예
214
제안된 데이터베이스에 테이블이 있는가?
아니오
218
현재 데이터베이스로부터 테이블을 탈락시킴
예
216
컬럼리스트의 각각의 테이블에 대해, DO:
단계 308로
(FIG.9E.)
220
유형 그룹컬럼의 컬럼인가?
예
224
컬럼리스트의 각각의 엘리먼트에 대해, 현재 스키마의 컬럼과 제안된 스키마의 컬럼을 비교
아니오
A

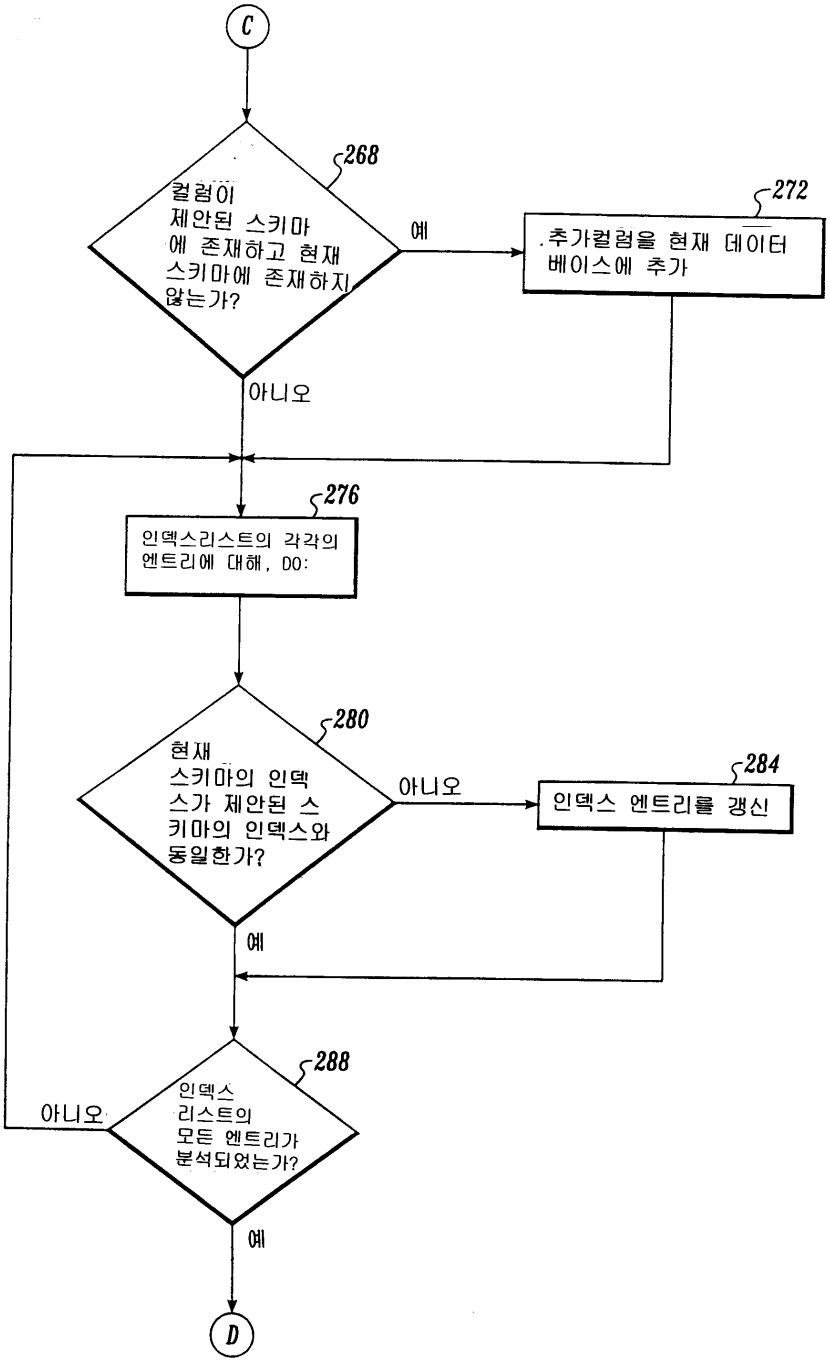
도면9B


A
228
컬럼이 유형 데이터 컬럼 또는 키컬럼인가?
아니오
예
232
제안된 스키마에서 현재 스키마의 컬럼이 발견되었는가?
아니오
234
데이터 컬럼 유형인 경우, 도 13으로 가고, 데이터 유형으로서 지정
예
236
테이블로부터 컬럼을 삭제
240
제안된 스키마로서 현재 스키마의 컬럼에 대해 데이터 유형, 논리길이, 비허용, 컬럼이름, 데이터길이 및 스케일 특성이 동일한가?
예
아니오
244
데이터 베이스의 컬럼의 특성을 갱신
B

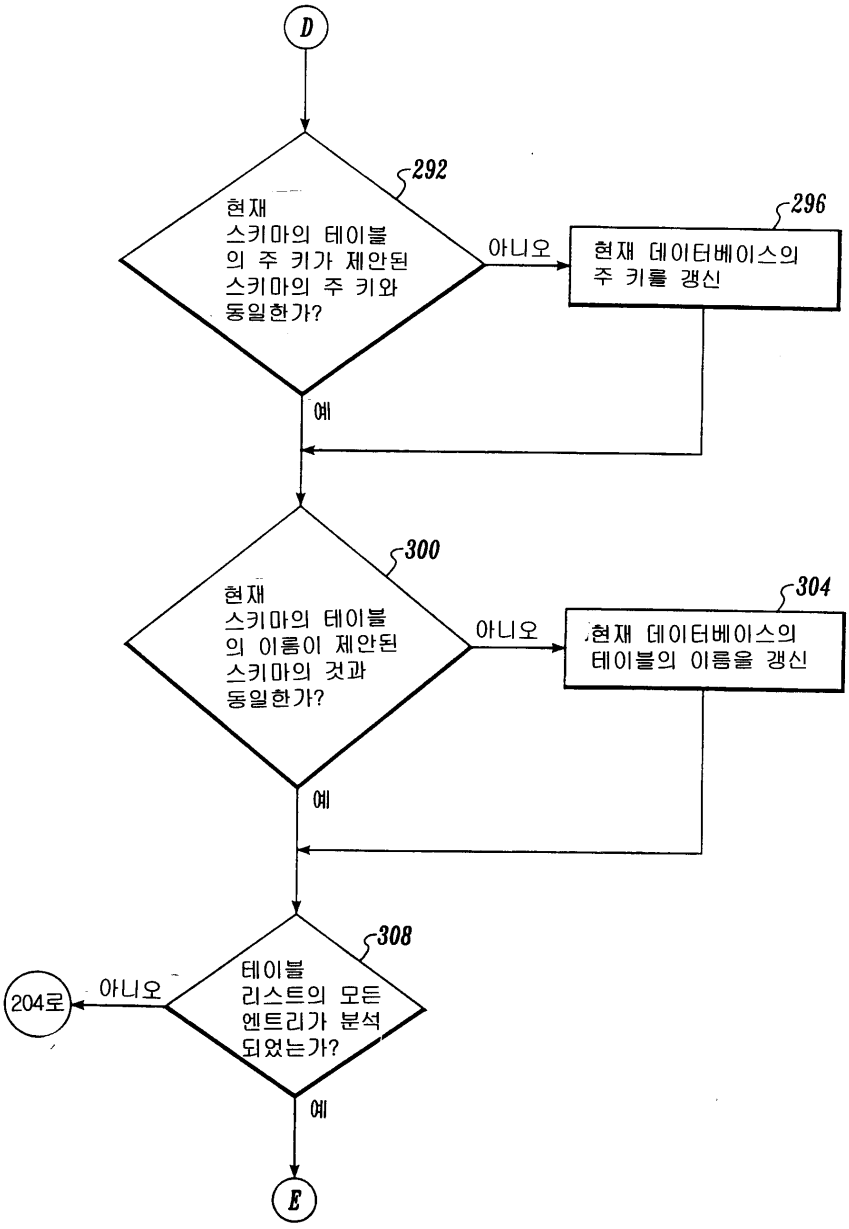
도면9C


B
248
외부키 유형 컬럼 인가?
아니오
예
252
제안된 스키마에 컬럼이 있는가 ?
아니오
254
도 13으로 감. 링크 유형으로 지정
예
260
현재 및 제안된 스키마의 외부키를 비교 및 갱신
256
현재 데이터베이스의 테이블로부터 외부키를 탈락시킴
264
컬럼 리스트의 모든 컬럼이 분석되었는가?
아니오
TO 216
예
C

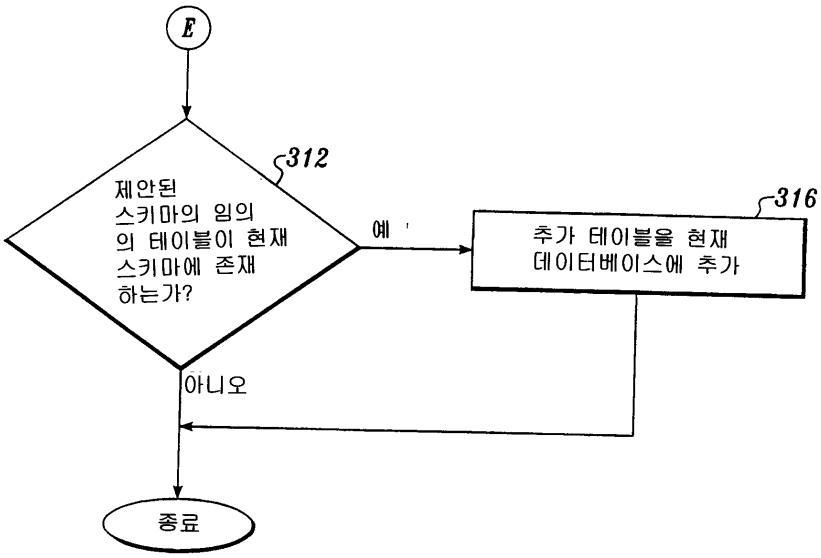
도면90


C
268
컬럼이 제안된 스키마에 존재하고 현재 스키마에 존재하지 않는가?
예
272
추가컬럼을 현재 데이터베이스에 추가
아니오
276
인덱스리스트의 각각의 엔트리에 대해, DO:
280
현재 스키마의 인덱스가 제안된 스키마의 인덱스와 동일한가?
아니오
284
인덱스 엔트리를 갱신
예
288
인덱스 리스트의 모든 엔트리가 분석되었는가?
아니오
예
D

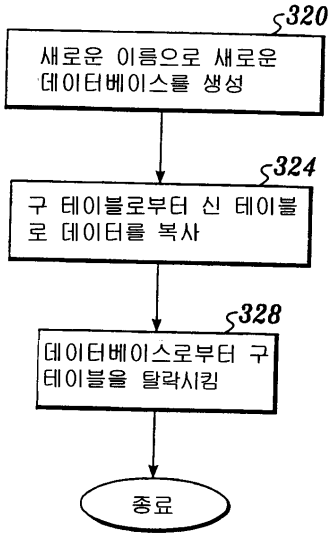
도면9E


D
292
현재 스키마의 테이블의 주 키가 제안된 스키마의 주 키와 동일한가?
아니오
296
현재 데이터베이스의 주 키를 갱신
예
300
현재 스키마의 테이블의 이름이 제안된 스키마의 것과 동일한가?
아니오
304
현재 데이터베이스의 테이블의 이름을 갱신
예
308
테이블 리스트의 모든 엔트리가 분석되었는가?
아니오
204로
예
E

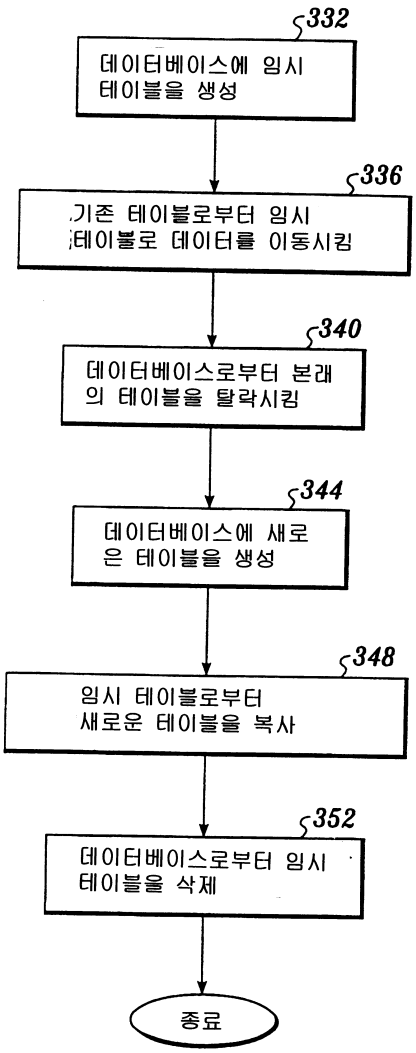
도면9F


E
312
제안된 스키마의 임의의 테이블이 현재 스키마에 존재하는가?
예
316
추가 테이블을 현재 데이터베이스에 추가
아니오
종료

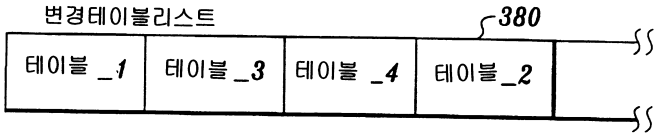
도면10A

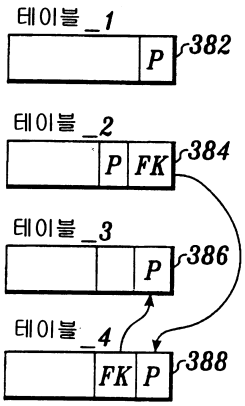

320
새로운 이름으로 새로운 데이터베이스를 생성
324
구 테이블로부터 신 테이블로 데이터를 복사
328
데이터베이스로부터 구 테이블을 탈락시킴
종료

도면10B

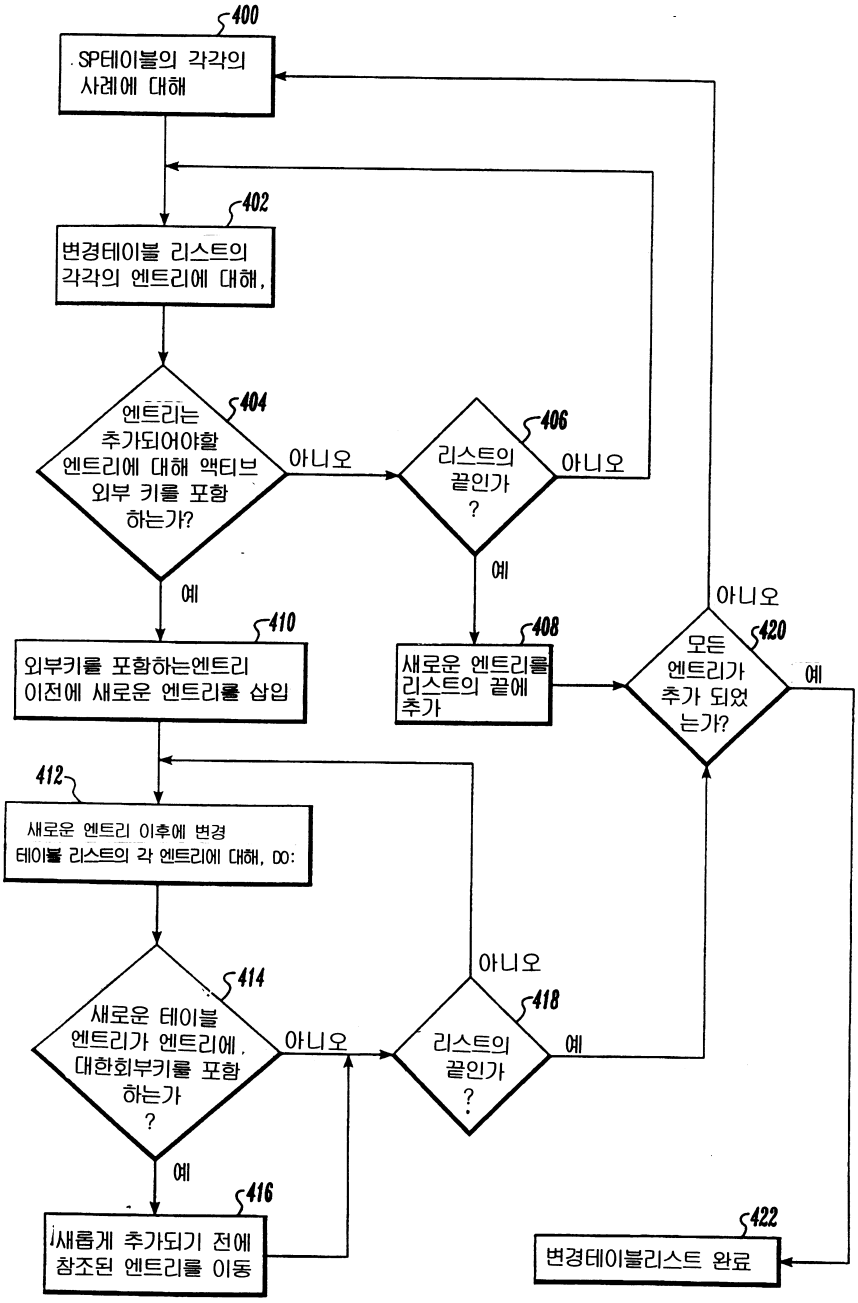

332
데이터베이스에 임시 테이블을 생성
336
기존 테이블로부터 임시 테이블로 데이터를 이동시킴
340
데이터베이스로부터 본래의 테이블을 탈락시킴
344
데이터베이스에 새로운 테이블을 생성
348
임시 테이블로부터 새로운 테이블을 복사
352
데이터베이스로부터 임시 테이블을 삭제
종료

도면11A

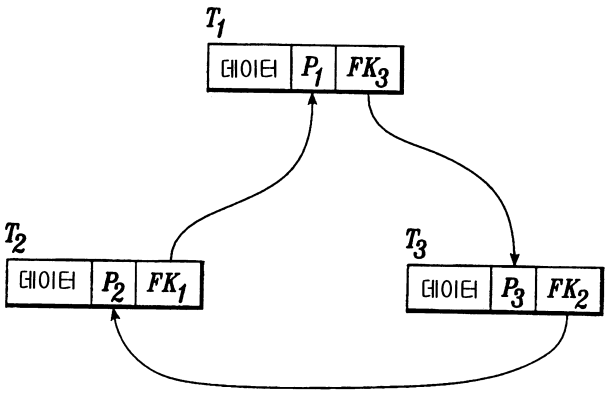

변경테이블리스트
380
테이블 _1
테이블 _3
테이블 _4
테이블 _2


테이블 _1
P
382
테이블 _2
P
FK
384
테이블 _3
P
386
테이블 _4
FK
P
388

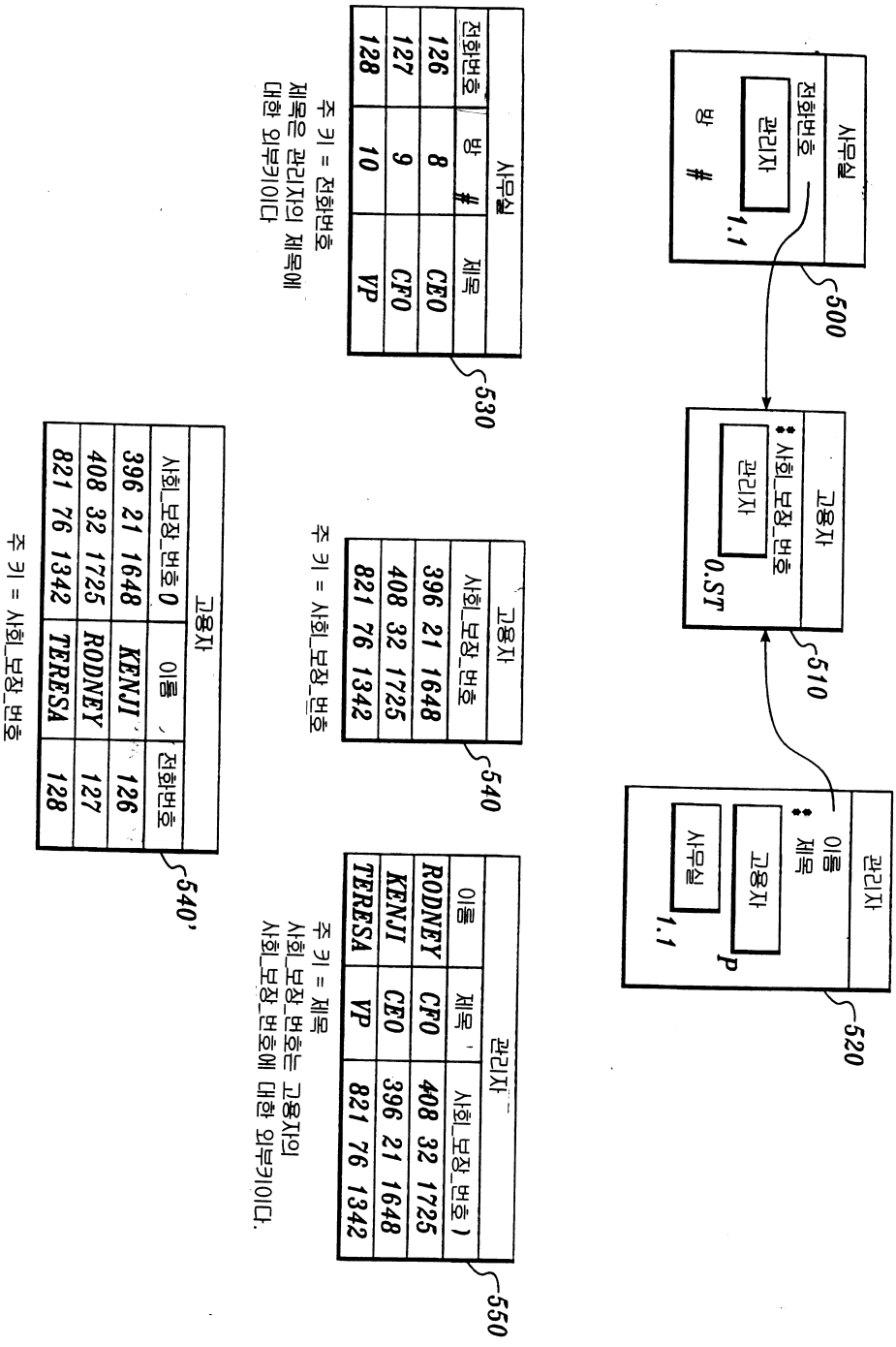
도면11B


400
SP테이블의 각각의 사례에 대해
402
변경테이블 리스트의 각각의 엔트리에 대해,
404
엔트리는 추가되어야할 엔트리에 대해 액티브 외부 키를 포함 하는가?
아니오
406
리스트의 끝인가 ?
아니오
예
예
410
외부키를 포함하는엔트리 이전에 새로운 엔트리를 삽입
408
새로운 엔트리를 리스트의 끝에 추가
아니오
420
모든 엔트리가 추가 되었 는가?
예
412
새로운 엔트리 이후에 변경 테이블 리스트의 각 엔트리에 대해, DO:
414
새로운 테이블 엔트리가 엔트리에 대한회부키를 포함 하는가 ?
아니오
418
리스트의 끝인가 ?
아니오
예
예
416
새롭게 추가되기 전에 참조된 엔트리를 이동
422
변경테이블리스트 완료

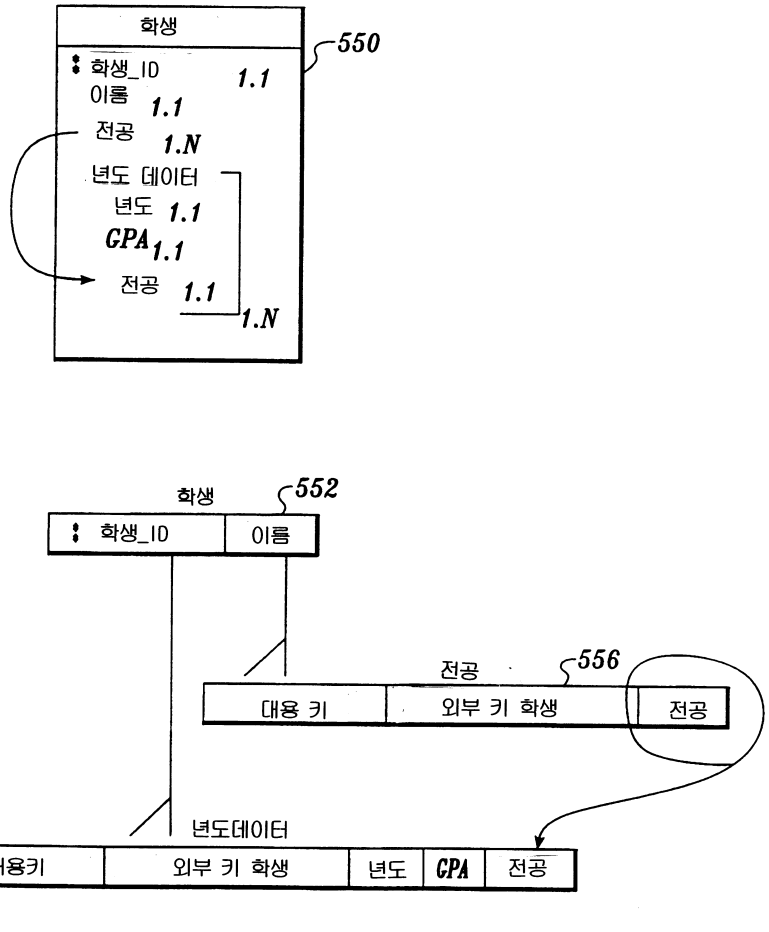
도면11C


T1
데이터
P1
FK3
T2
데이터
P2
FK1
T3
데이터
P3
FK2

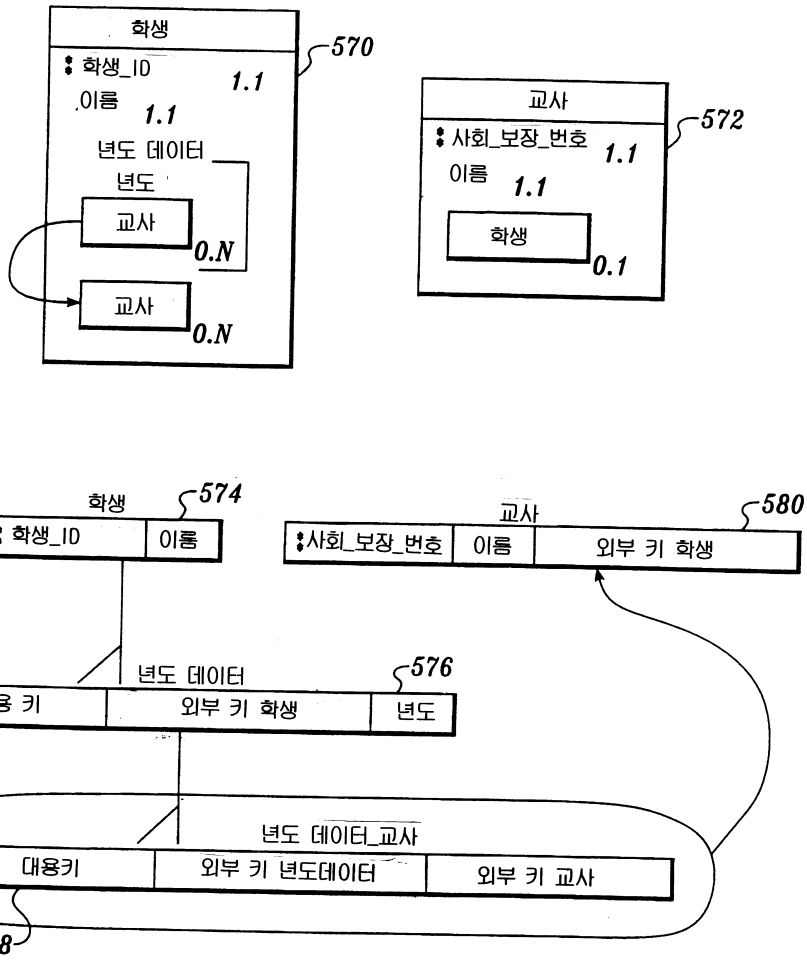
도면12


사무실
전화번호
관리자
방 #
1.1
500
고용자
사회_보장_번호
관리자
0.ST
510
관리자
이름
제목
고용자
사무실
1.1
P
520
사무실
전화번호
방 #
제목
126
8
CEO
127
9
CFO
128
10
VP
530
주 키 = 전화번호
제목은 관리자의 제목에
대한 외부키이다
고용자
사회_보장_번호
396 21 1648
408 32 1725
821 76 1342
540
주 키 = 사회_보장_번호
관리자
이름
제목
사회_보장_번호)
RODNEY
CFO
408 32 1725
KENJI
CEO
396 21 1648
TERESA
VP
821 76 1342
550
주 키 = 제목
사회_보장_번호는 고용자의
사회_보장_번호에 대한 외부키이다.
고용자
사회_보장_번호 0
이름
전화번호
396 21 1648
KENJI
126
408 32 1725
RODNEY
127
821 76 1342
TERESA
128
540'
주 키 = 사회_보장_번호

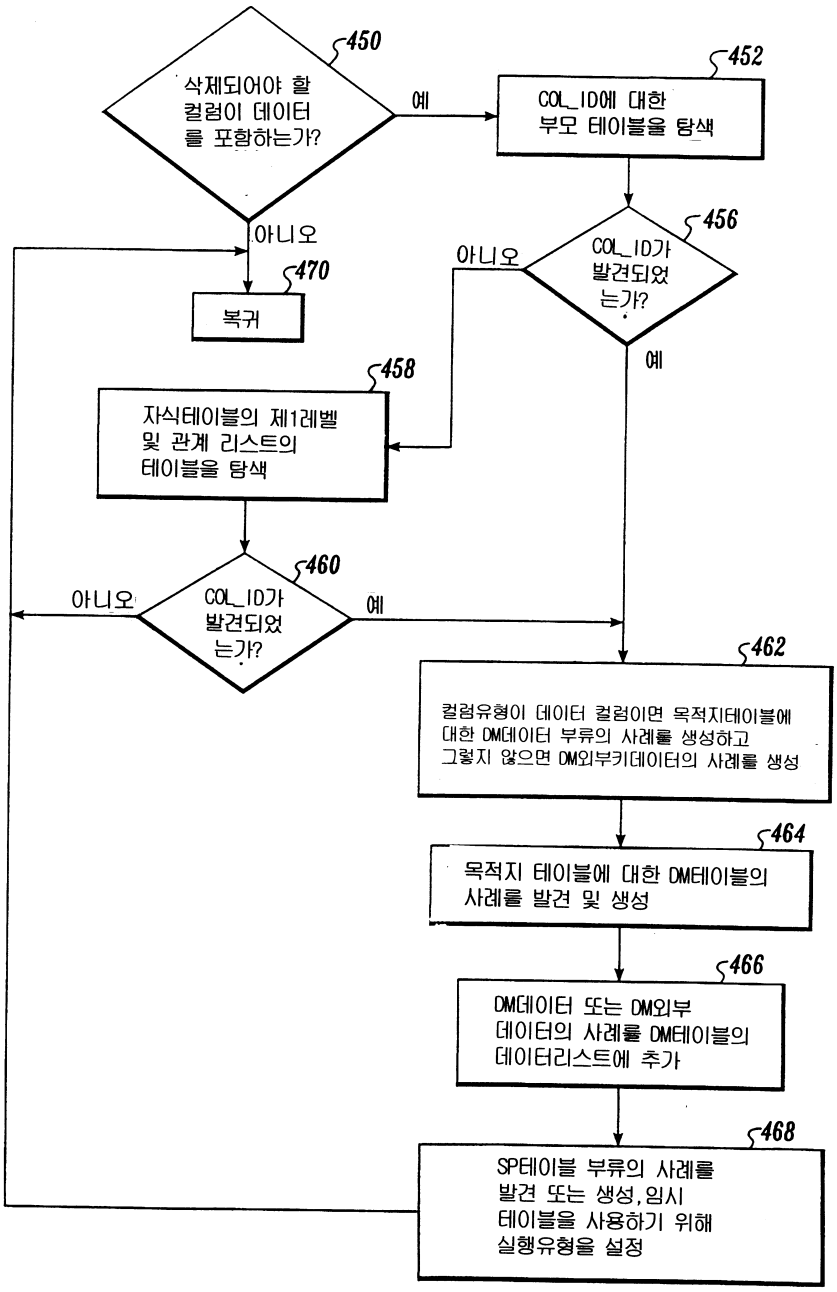
도면12A


학생
550
학생_ID
1.1
이름
1.1
전공
1.N
년도 데이터
년도
1.1
GPA
1.1
전공
1.1
1.N
학생
552
학생_ID
이름
전공
556
대용 키
외부 키 학생
전공
년도데이터
대용키
외부 키 학생
년도
GPA
전공
554

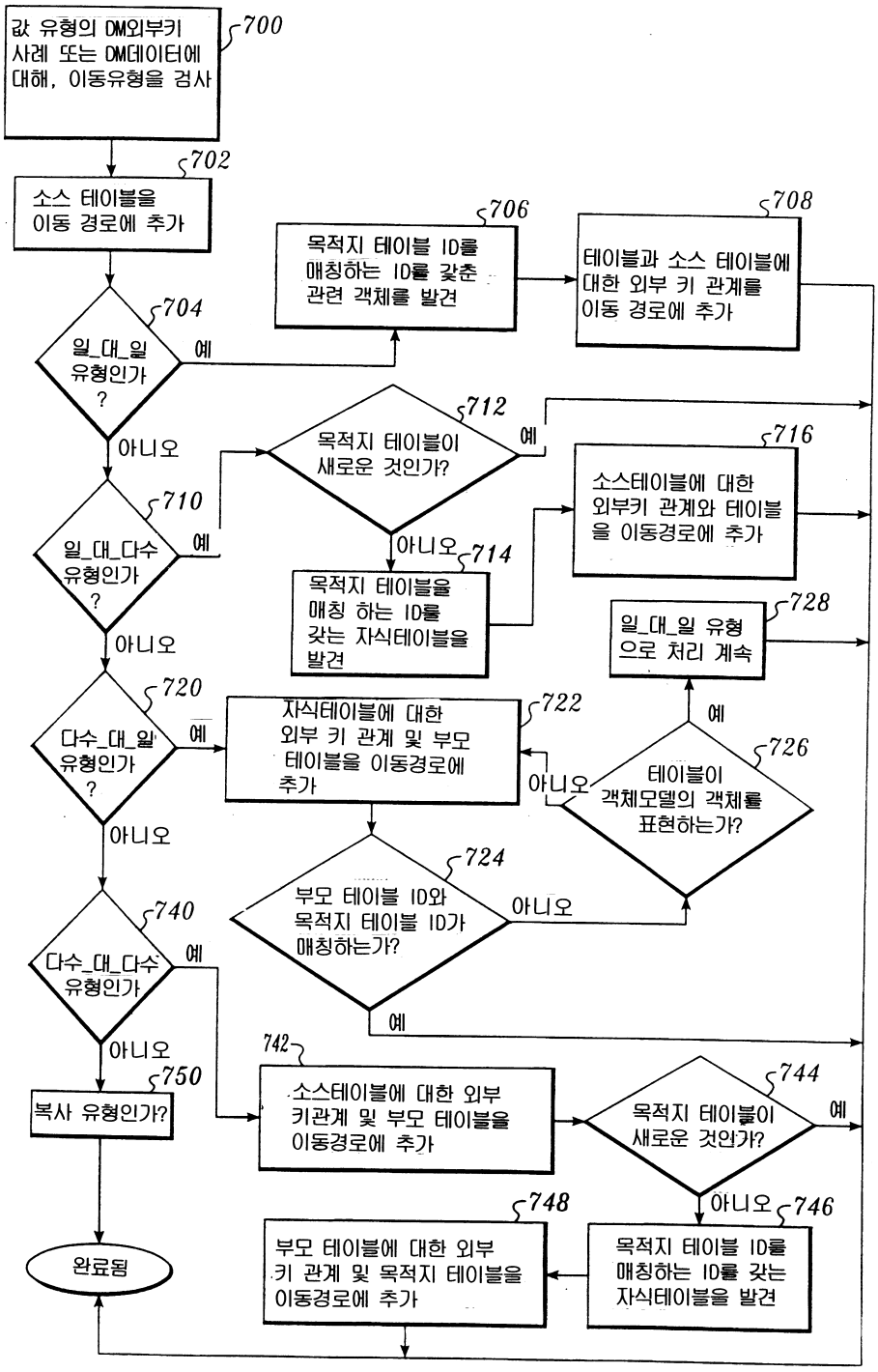
도면128


학생
학생_ID
1.1
이름
1.1
년도 데이터
년도
교사
0.N
교사
0.N
570
교사
사회_보장_번호
1.1
이름
1.1
학생
0.1
572
학생
574
학생_ID
이름
교사
580
사회_보장_번호
이름
외부 키 학생
년도 데이터
576
대용 키
외부 키 학생
년도
년도 데이터_교사
대용키
외부 키 년도데이터
외부 키 교사
578

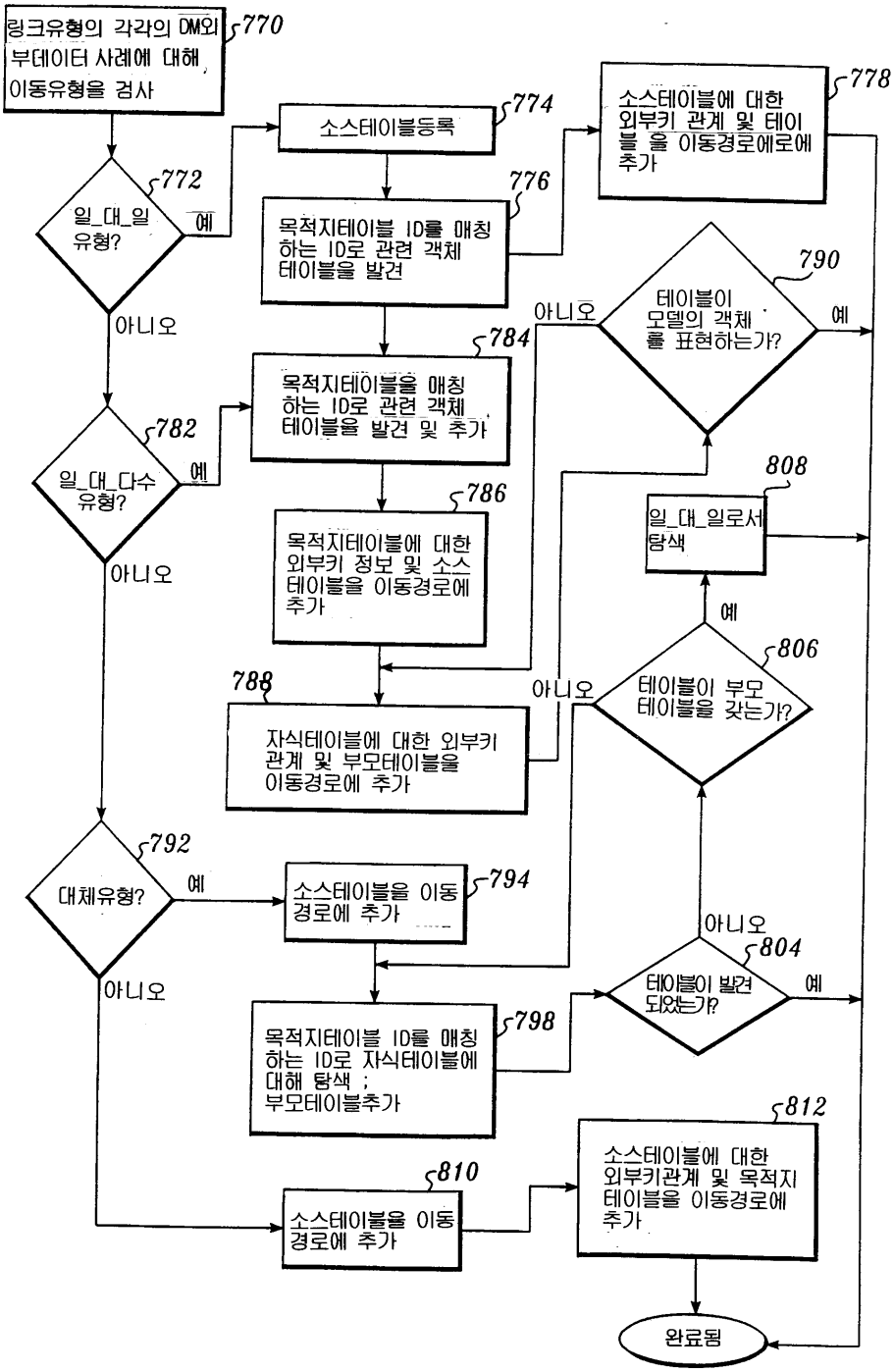
도면13


450
삭제되어야 할 컬럼이 데이터를 포함하는가?
예
452
COL_ID에 대한 부모 테이블을 탐색
아니오
456
COL_ID가 발견되었는가?
아니오
예
470
복귀
458
자식테이블의 제1레벨 및 관계 리스트의 테이블을 탐색
460
COL_ID가 발견되었는가?
아니오
예
462
컬럼유형이 데이터 컬럼이면 목적지테이블에 대한 DM데이터 부류의 사례를 생성하고 그렇지 않으면 DM외부키데이터의 사례를 생성
464
목적지 테이블에 대한 DM테이블의 사례를 발견 및 생성
466
DM데이터 또는 DM외부 데이터의 사례를 DM테이블의 데이터리스트에 추가
468
SP테이블 부류의 사례를 발견 또는 생성,임시 테이블을 사용하기 위해 실행유형을 설정

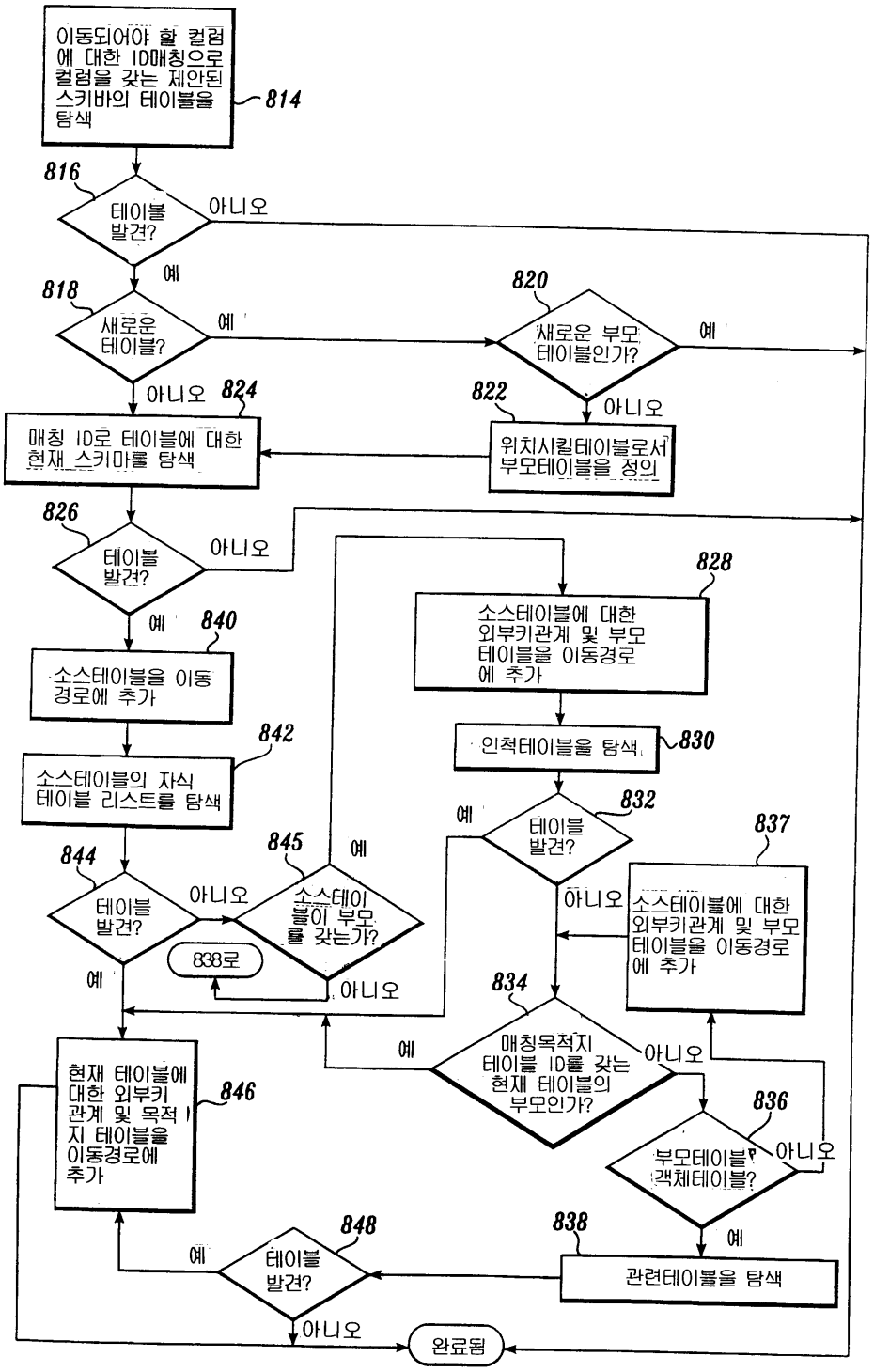
도면14A


값 유형의 OM외부키 사례 또는 OM데이터에 대해, 이동유형을 검사
700
소스 테이블을 이동 경로에 추가
702
일_대_일 유형인가 ?
704
예
목적지 테이블 ID를 매칭하는 ID를 갖춘 관련 객체를 발견
706
테이블과 소스 테이블에 대한 외부 키 관계를 이동 경로에 추가
708
아니오
일_대_다수 유형인가 ?
710
예
목적지 테이블이 새로운 것인가?
712
예
아니오
목적지 테이블을 매칭 하는 ID를 갖는 자식테이블을 발견
714
소스테이블에 대한 외부키 관계와 테이블을 이동경로에 추가
716
아니오
다수_대_일 유형인가 ?
720
예
자식테이블에 대한 외부 키 관계 및 부모 테이블을 이동경로에 추가
722
일_대_일 유형으로 처리 계속
728
예
테이블이 객체모델의 객체를 표현하는가?
726
아니오
아니오
다수_대_다수 유형인가
740
예
부모 테이블 ID와 목적지 테이블 ID가 매칭하는가?
724
아니오
예
아니오
복사 유형인가?
750
742
소스테이블에 대한 외부 키관계 및 부모 테이블을 이동경로에 추가
목적지 테이블이 새로운 것인가?
744
예
아니오
746
목적지 테이블 ID를 매칭하는 ID를 갖는 자식테이블을 발견
748
부모 테이블에 대한 외부 키 관계 및 목적지 테이블을 이동경로에 추가
완료됨

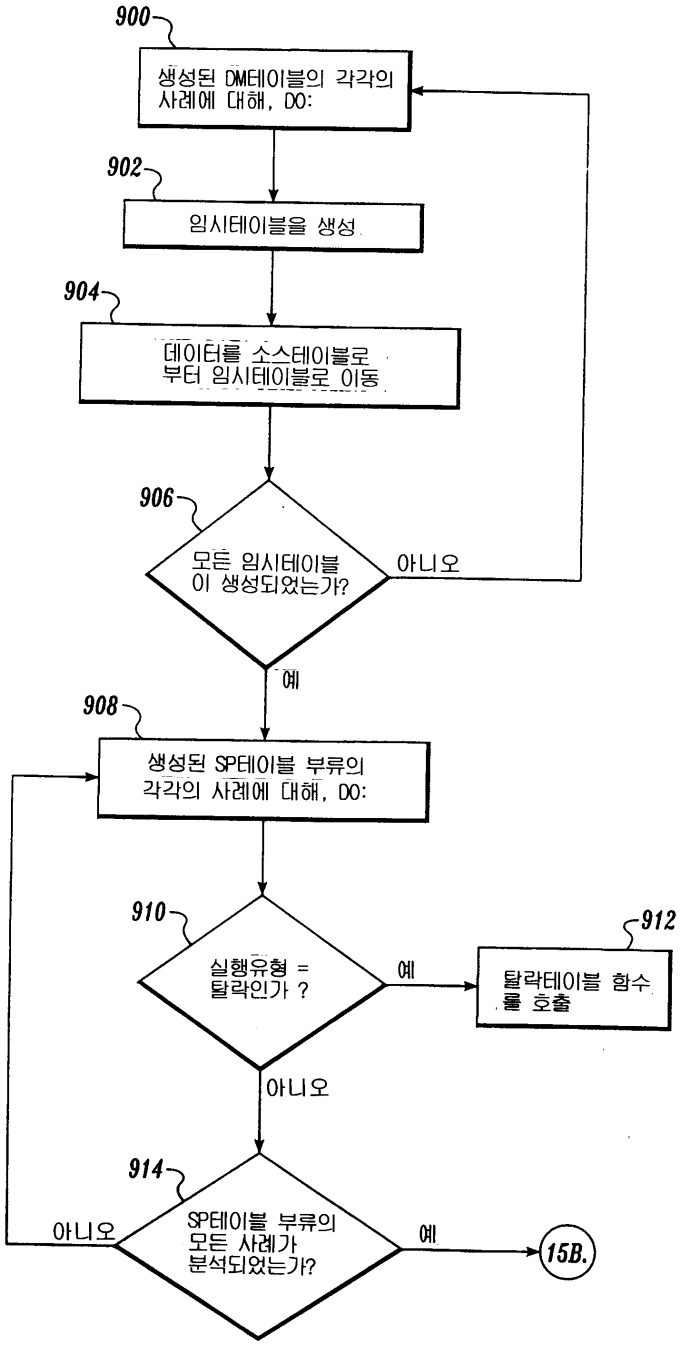
도면14B


링크유형의 각각의 DM외 부데이터 사례에 대해 이동유형을 검사
770
일_대_일 유형?
772
예
아니오
소스테이블등록
774
목적지테이블 ID를 매칭하는 ID로 관련 객체 테이블을 발견
776
소스테이블에 대한 외부키 관계 및 테이블 을 이동경로에로에 추가
778
테이블이 모델의 객체를 표현하는가?
790
예
아니오
일_대_다수 유형?
782
예
아니오
목적지테이블을 매칭하는 ID로 관련 객체 테이블을 발견 및 추가
784
목적지테이블에 대한 외부키 정보 및 소스 테이블을 이동경로에 추가
786
일_대_일로서 탐색
808
예
테이블이 부모 테이블을 갖는가?
806
아니오
자식테이블에 대한 외부키 관계 및 부모테이블을 이동경로에 추가
788
대체유형?
792
예
아니오
소스테이블을 이동경로에 추가
794
아니오
테이블이 발견되었는가?
804
예
목적지테이블 ID를 매칭하는 ID로 자식테이블에 대해 탐색 ; 부모테이블추가
798
소스테이블을 이동경로에 추가
810
소스테이블에 대한 외부키관계 및 목적지 테이블을 이동경로에 추가
812
완료됨

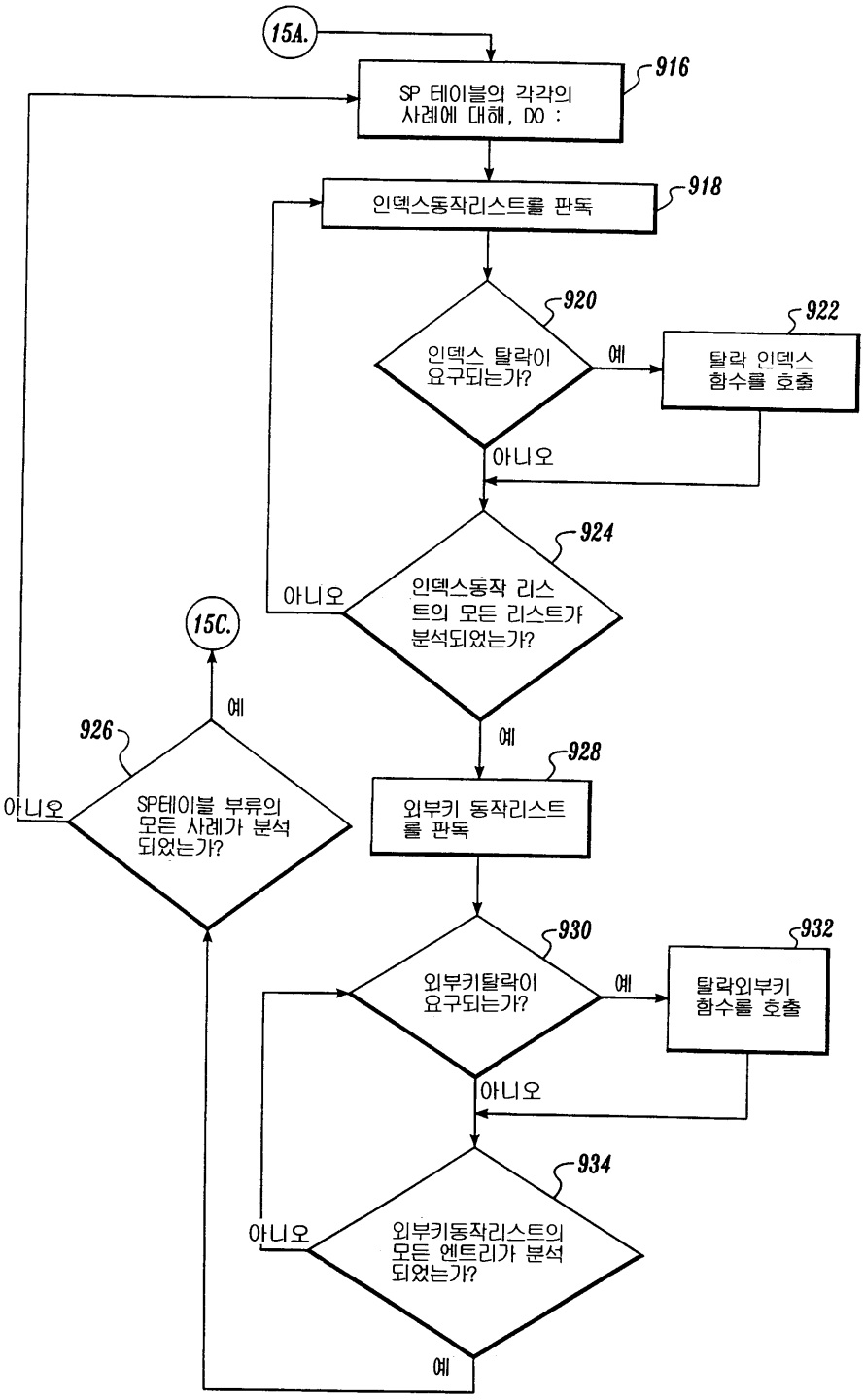
도면14C


이동되어야 할 컬럼에 대한 ID매칭으로 컬럼을 갖는 제안된 스키바의 테이블을 탐색
814
816
테이블 발견?
아니오
예
818
새로운 테이블?
예
820
새로운 부모 테이블인가?
예
아니오
824
822
아니오
매칭 ID로 테이블에 대한 현재 스키마를 탐색
위치시킬테이블로서 부모테이블을 정의
826
테이블 발견?
아니오
예
840
소스테이블을 이동 경로에 추가
828
소스테이블에 대한 외부키관계 및 부모 테이블을 이동경로에 추가
842
소스테이블의 자식 테이블 리스트를 탐색
인척테이블을 탐색
830
832
예
테이블 발견?
837
844
845
예
테이블 발견?
아니오
소스테이블이 부모를 갖는가?
아니오
838로
아니오
소스테이블에 대한 외부키관계 및 부모 테이블을 이동경로에 추가
예
834
예
매칭목적지 테이블 ID를 갖는 현재 테이블의 부모인가?
아니오
현재 테이블에 대한 외부키 관계 및 목적지 테이블을 이동경로에 추가
846
836
부모테이블? 객체테이블?
아니오
848
838
예
테이블 발견?
예
관련테이블을 탐색
아니오
완료됨

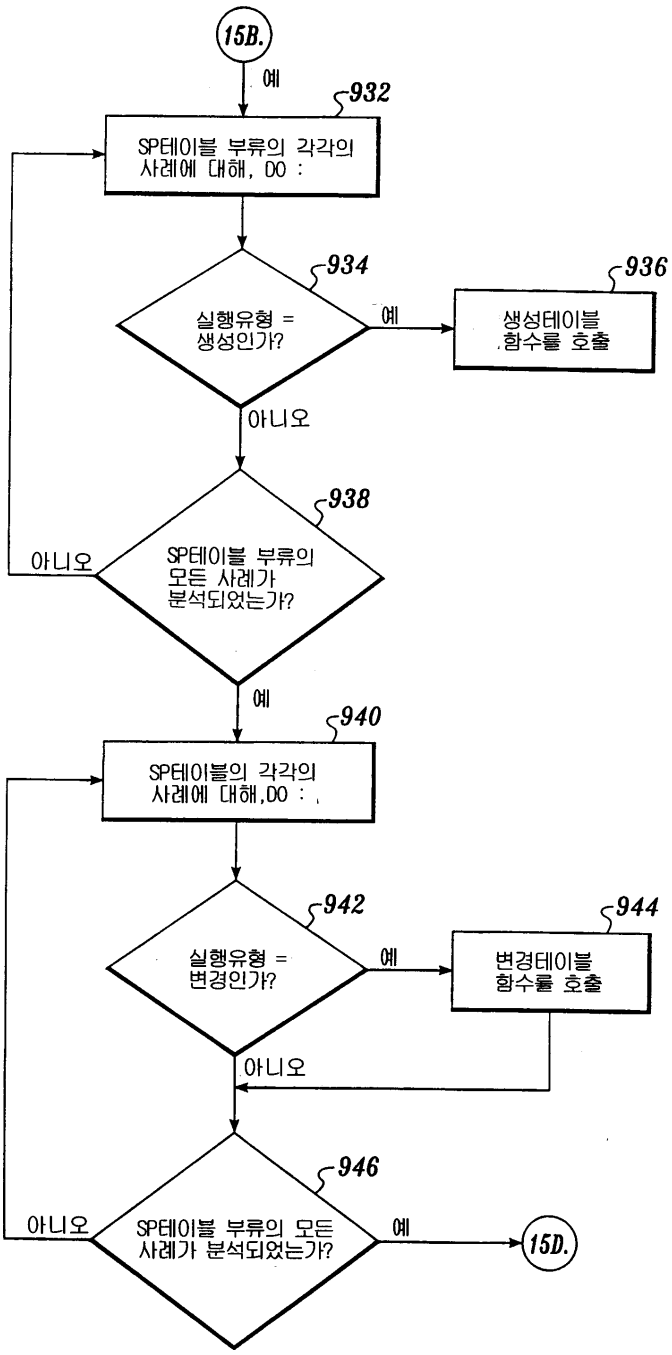
도면15A


900
생성된 DME테이블의 각각의 사례에 대해, DO:
902
임시테이블을 생성
904
데이터를 소스테이블로부터 임시테이블로 이동
906
모든 임시테이블이 생성되었는가?
아니오
예
908
생성된 SP테이블 부류의 각각의 사례에 대해, DO:
910
실행유형 = 탈락인가 ?
예
912
탈락테이블 함수를 호출
아니오
914
SP테이블 부류의 모든 사례가 분석되었는가?
아니오
예
15B.

도면15B


15A.
916
SP 테이블의 각각의 사례에 대해, DO :
918
인덱스동작리스트를 판독
920
인덱스 탈락이 요구되는가?
예
922
탈락 인덱스 함수를 호출
아니오
924
인덱스동작 리스트의 모든 리스트가 분석되었는가?
아니오
15C.
예
926
SP테이블 부류의 모든 사례가 분석 되었는가?
아니오
예
928
외부키 동작리스트를 판독
930
외부키탈락이 요구되는가?
예
932
탈락외부키 함수를 호출
아니오
934
외부키동작리스트의 모든 엔트리가 분석 되었는가?
아니오
예

도면15C


15B.
예
932
SP테이블 부류의 각각의 사례에 대해, DO :
934
실행유형 = 생성인가?
예
936
생성테이블 함수를 호출
아니오
938
아니오
SP테이블 부류의 모든 사례가 분석되었는가?
예
940
SP테이블의 각각의 사례에 대해,DO :
942
실행유형 = 변경인가?
예
944
변경테이블 함수를 호출
아니오
946
아니오
SP테이블 부류의 모든 사례가 분석되었는가?
예
15D.