전문가 리포트

이질데이터베이스 통합 (Ⅱ)

Integration of Heterogeneous Distributed Databases

나민영/육군사관학교 전산학교수
Ra, Min-Young. Ph.D. Assistant Professor
Dept. of Mathematics & Computer Science
Korea Military Academy

2 스키마 변환 및 통합

이 장에서는 이질 데이터베이스 통합에 있어 가장 핵심 기술중 하나인 스키마 변환 및 통합 기법을 다룬다.

21 스키마 변환

스키마 변환은 입력스키마로부터 출력스키마를 얻는 작업으로서 1장에서 살펴본 바와 같이 이질 데이터베이스 통합에 가장 기초가 되는 작업이다. 먼저 스키마 변환의 종류를 살펴보면 다음과 같다.

● Minimality를 위한 변환

스키마의 잉여는 관계성의 사이클, derived 애트리뷰트, implicit 서브셋 등의 이유 때문에 생긴다. 잉여를 그대로 두던지 혹은 제거하든지 하는 것은 설계자에게 달려 있으나 잉여는 보통 데이터베이스 관리상 이상현상(anomaly)의 원인이 되므로 제거한다. 잉여의 제거는 스키마변환을 통해 이루어진다. 예를 들어 그림 21은 관계성의 사이클로 인한 잉여를 보여주고 있다. 이 그림에서 works-with는 works-in 과 heads로 된 path와 동등하므로 제거해도 된다.

● Expressiveness (표현성) 증가를 위한 변환

스키마의 표현성은 스키마를 간단히 함으로서 고양된다. 표현성을 증가시키기 위한 전형적인 스키마변환은 먼저 일반화 계층에서 dangling 서브엔티티 및 엔티티를 제거하고 일반화 개념을 이용해서 가능한한 일반화를 함으로서 이루어진다. 예를 들어 그림 22는 일반화 계층구조에서 불필요한 서브타입을 제거하여 단순화된 구조를 보여주고 있다. 즉, 애트리뷰트 RANK의 도메인 값이 (PROFESSOR, INSTRUCTOR, GRADU-

ATE__STUDENT) 인 하나의 엔티티로 간단히 표현될 수 있다. 또한 그림 23은 dangling 엔티티가 제거될 수 있음을 보여준다. 즉, CITY__OF__BIRTH는 엔티티로 표현되어 있지만 이는 PERSON의 단순 애트리뷰트로 종합되어질 수 있다.

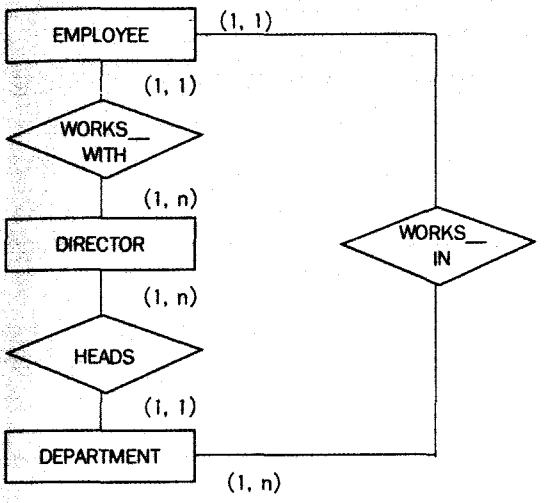


그림 21 관계성의 사이클로 인한 잉여

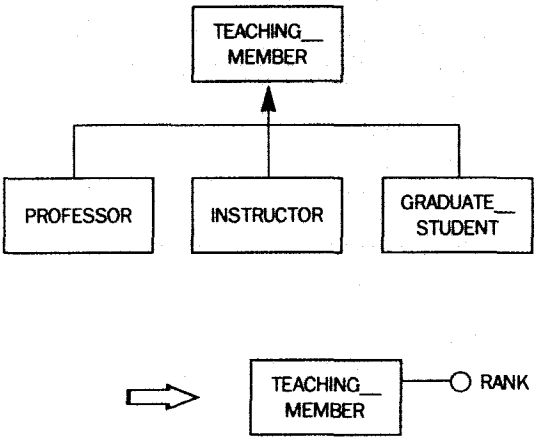


그림 22 일반화 계층의 단순화

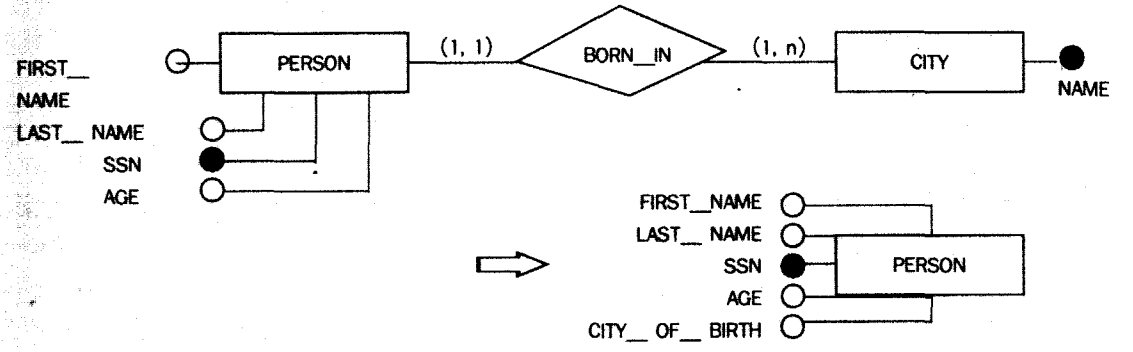


그림 23 Dangling 엔티티의 제거

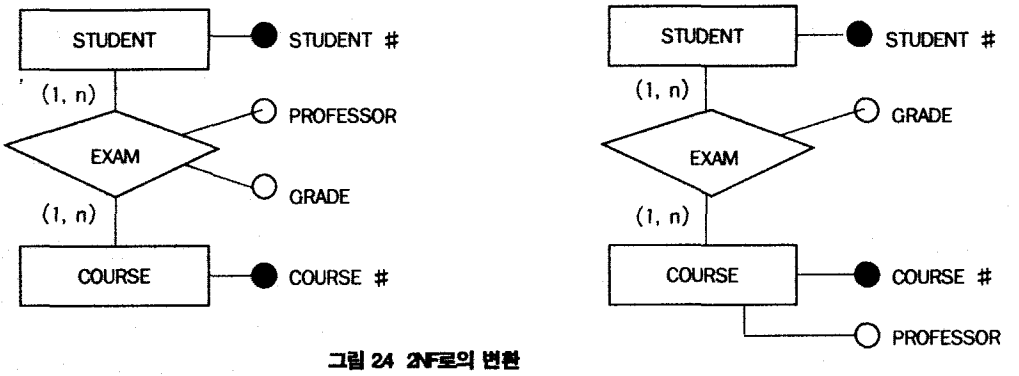


그림 24 2NF로의 변환

● Normalization을 위한 변환

관계형 모델에서 릴레이션의 정규화는 원하는 정규형을 얻기 위해 점진적 변환을 적용하는 과정이다. 이 과정은 통상 함수적 종속성(functional dependency)을 분석함으로서 2NF, 3NF, BCNF 등의 정규형으로 발전해 감으로서 이루어진다. 정규형으로 스키마변환이 되어 있지 않으면 갱신시 삽입이상, 삭제이상, 갱신이상 등의 이상현상으로 데이터베이스의 일치를 기대하기가 어렵다. 그림 24는 정규형 변환중 2NF를 얻기 위한 변환을 보여주고 있다.

● 이질 모델 액세싱을 위한 변환

지금까지 앞에서 다룬 변환은 한 스키마의 질을 높이기 위해 수행하는 변환이었다. 그러나 한 사용자가 다른 모델의 데이터베이스를 액세스하려고 할때에는 원시스키마로부터 목표스키마로 스키마변환이 이루어져야 한다. 예를 들어 CDM이 EER 모델이고 통합될 참여 DBMS의 하나는 관계형이고 다른 하나는 CODASYL이며 연합사용자들이 관계형 뷰를 원한다고 하자. 그러면 (1) 관계형 스키마의 지역스키마를 EER로 표현된 동등한 구성스키마로 변환하는 작업, (2) EER 모델의 연합스키마의 일부를 관계형 모델로 표현된 외부스키마로 변환하는 작업 등이 요구된다.

CDM과 같은 시맨틱 데이터 모델의 사용은 관계형과 같은 재래식 모델에서는 명시하기 곤란한 부가적 시맨틱의 표현을 쉽게해줄 수 있다. 예를 들어 두 엔티티 타입간 inheritance를 허용하는 일반화 관계성을 시맨틱 데이터모델을 사용하는 구성스키마에서는 명백하게 표현할 수 있다. 다른 예로서 관계형 모델로 표현된 지역스키마에서 릴레이션 R1과 R2간의 외부키는 대응되는 EER 모델의 구성스키마에서 R1과 R2를 나타내는 두 엔티티간의 관계성으로서 명백히 표현될 수 있다.

22 이질 데이터베이스 액세싱을 위한 스키마변환

이질 데이터베이스 액세싱을 위한 스키마 변환은 ER모델과 O_O모델등 특정된 일부 모델간에서 비교적 활발하게 이루어져 왔다.

이 절에서는 이질 데이터베이스 통합을 위한 연구중 최근 진행중인 연구를 소개한다. 이 연구들은 1장에서 소개된 이질 데이터베이스 통합 접근방법중 가장 강력하고 편리한 기법인 multimodel multilingual 접근 방법에 근거한 연구들이다.

22.1 M(DM)

M(DM)은 이질 데이터 및 이질 데이터 표현에 따른 문제점들을 해결하기 위해 IBM Watson 연구소에서 1992년에 제안한 Extensible Meta System이다. 모델간 이질성을 다루기 위한 확장형 메타 시스템의 사용은 공통 작업용으로서 하나의 데이터 모델이 필요했던점을 제거할 수 있다. 더우기 재래식 방법은 서로 다른 모델들이 매핑될 수 있는 더 표현력이 풍부한 모델을 지속적으로 요구하지 않을 수 없다. 그러나 하나의 데이터 모델은 이러한 계속적인 요구를 만족시킬 수가 없다. 실제적인 상호작용 측면에서 M(DM)은 스키마변환, 다중 모델 질의어를 통하여 데이터베이스 투명성을 제공한다.

데이터베이스 투명이 어떻게 이루어지는가를 살펴보면 다음과 같다. M(DM) 연합은 각각의 스키마, DBMS, 데이터 모델과 함께 여러 자치적인 지역 데이터베이스들로 구성된다. 이때 각 지역 데이터베이스는 서브스키마를 명시하거나 혹은 지역데이터베이스의 공유 부분인 수출스키마(export schema)를 명시한다. 물론 수출스키마는 현재까지는 각각의 데이터 모델로 표현된다. 다음 단계는 수출스키마를 동등한 M(DM) 수입스키마 (import schema)로 전환시키는 것이다. 이

수입스키마들이 모여서 M(DM) 멀티데이터베이스 스키마를 정의한다. 이러한 프로세스는 DBMS 독립을 낳는다. 즉, 사용자는 하나의 언어 (즉 M(DM)-L)를 이용하여 멀티플 데이터베이스를 액세스할 수 있게 되고 각각의 데이터 모델과 질의어의 특이한 개성들을 다룰 필요가 없어진다. 마지막 단계는 통합된 뷰가 이 M(DM)상에 정의되어 멀티플 데이터베이스의 존재를 완전히 가릴 수 있다. 따라서 데이터베이스 투명성을 제공할 수 있고 사용자가 정의한 뷰에 대한 M(DM)-L 질의어가 각 장소에서 투명하게 수행될 수 있다.

그러나 M(DM)은 새로운 모델이 탄생될 때마다 이 모델을 표현하기 위한 타입을 계속 추가해야 하는 번거로움을 가지고 있다.

222 트리 형태 중간구조

지금까지의 연구는 통상 통합 스키마의 표현에 사용될 개념스키마를 결정한후 각 지역스키마를 결정된 개념스키마로 변환하여 사용하거나 또는 이들을 통합하여 사용하였다.

그러나 이러한 방법은 한 모델에서 다른 모델로 완전한 변환을 해야 하므로 결정된 개념스키마에 의존하게 되고 스키마 변환 중간 과정에서 통합하는 것보다 더 많은 오버헤드(overhead)를 야기한다. Openbase 시스템에서는 트리 형태의 중간내부구조를 이용한 효과적인 스키마 변환 기법을 사용중에 있다.

Openbase 시스템의 사용자는 스키마 변환을 자신의 스키마로 다른 데이터베이스를 보는데 사용하게 된다. 따라서 CDM을 이용하여 일단 네트워크에 가입한 모든 원시스키마를 각각 CDM으로 변환 저장해놓고 필요시 이를 해당되는 목표스키마로 변환하여 사용하게 된다. 이를 개념적으로 나타내면 그림 25(a)와 같다. 여기서 CDM이란 스키마 변환이나 통합시 데이터 모델 및 DBMS의 다양성 때문에 발생하는 문제들을 해결하기 위해서 사용되는 공통 데이터 모델 (Common Data Model)로서 이를 이용하여 원시스키마인 각 지역스키마들이 표현된다. 이때 사용되는 CDM으로서는 ER 모델이나 관계형 모델과 같은 완전한 모델을 이용하는 방법과 캐노니칼 형태의 중간구조를 이용하는 방법이 있을 수 있다.

그러나 CDM으로 개념적 모델과 같은 완전한 모델을 사용하면 한 모델에서 다른 모델로 완전한 변환을 해야 하므로 결정된 CDM에 의존하게 된다. 따라서 원시스키마들이 스키마변환 중간과정에서 스키마의 시맨틱을 최대로 유지할 수 있는 중간구조의 형태로 유지되면 훨씬 효과적일 것이다. 그림 25(b)는 이러한 환경을 보여준다.

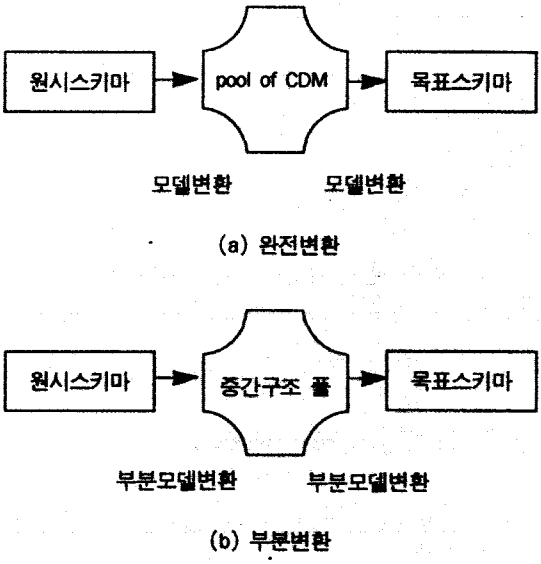


그림 25 OpenBase 시스템에서의 스키마변환

중간구조로서 OpenBase에서는 이진 트리형태의 리스트 구조를 갖는다. 트리를 이용하여 제안된 중간구조는 사물을 표현하기 위한 방법으로 래티스(lattice) 개념을 이용한다. 래티스 계층에는 다음 그림 26에서 보는 바와 같이 네가지 타입이 있다. 최상위 계층은 'Thing'으로서 현실 세계에 있는 것은 무엇이든지 해당된다. 'Thing'은

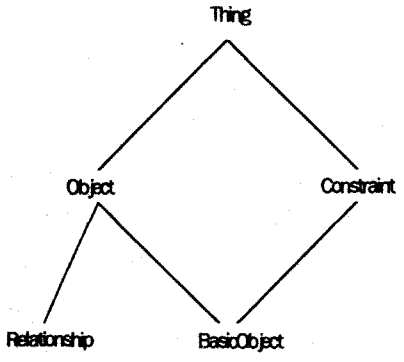


그림 26 레티스 계층구조

'Object'와 'Constraint'로 구분된다. 'Object'는 존재하는 어떤것을 가리키고 이는 다시 'Basic Object'와 'Relationship'으로 구분된다.

'BasicObject'는 'Object'의 서브타입으로서 원자값을 갖는 객체를 나타낸다. 'Relationship'은 Object와 Object간 또는 Object와 BasicObject간의 1:1, 1:N, M:N 관계를 나타낸다. 'Constraint'는 Object의 시맨틱, 예를 들면 서브셋 관계, 키 제약조건, 다중값 에트리뷰트 등을 나타낸다. 위 그림에서 모든 BasicObject는 Object임을 이해하는 것은 그리 어렵지 않다.

23 스키마 통합 (Schema Integration)

스키마 통합이란 이미 존재하는 또는 제안된 데이터 베이스들을 하나의 통일된 전역 스키마로 합하는 행위이다. 스키마 통합은 다음과 같은 두 가지 관점에서 생각해 볼 수 있다.

뷰 통합(데이터베이스 설계시): 제안된 데이터베이스의 전체적인 개념적 묘사 (conceptual description)를 생성.

데이터베이스 통합 (분산 데이터베이스 관리시): 데이터베이스 모음에 대한 전역스키마 생성. 이 전역스키마는 분산 데이터베이스 환경에 참여한 모든 데이터베이스들의 가상 뷰이다.

여기서 다루는 스키마 통합이란 데이터베이스 통합과 관련된 것으로 이미 존재하는 이질 데이터베이스의 스키마들을 하나의 스키마로 통합하는 것을 말한다. 스키마통합에는 이미 많은 연구가 수행되어 왔다. 스키마를 통합할때 수행되는 작업에는 (1) 서로 다른 원시스키마 (또는 구성 스키마)에 있는 객체들이 어떻게 연관되어 있는가를 알아내고 결정하기 위한 스키마와 객체의 분석, (2) 원시스키마에 대한 매핑과 함께 통합된 객체를 생성하기 위한 스키마와 객체의 합병 등의 두가지 형태가 있다. 이러한 작업을 지원하기 위해 사용되는 테크닉은 스키마 분석과 통합의 목적, 스키마의 데이터 모델, 정보의 완전성, 통합기의 능력 등에 따라 좌우된다. 분석 작업에 대한 기술이 합병 작업에 대한 기술보다 더 어려운 것으로 간주된다. 또한 하나의 기법만으로는 충분하지 않고 여러 기술이 함께 사용된다.

이러한 스키마 통합 작업을 자동화하면 매우 편리할 것이다. 그러나 완전한 자동 스키마 통합이 불가능한 이유중 하나는 스키마의 모든 시맨틱이 완전히 명세될 수 없기 때문이다. 스키마의 모든 시맨틱이 완정히 명세될 수 없는 이유는 (1) 현재의 시맨틱 모델이 현실세계 상태를 완전히 나타낼 수 없기 때문에, (2) 통상 스키마에 내포된 정보보다 더 많은 정보를 필요로 하기 때문에, (3) 현실 세계 상태에 대한 해석 및 관점이 다양하기 때문에 등이다.

23.1 통합절차

본 절에서는 Sheth의 방법과 Batini et al.의 방법을 살펴보기로 한다.

(1) Sheth의 통합 기법

단계 1 : 프레통합 (Preintegration)

여러 모델을 통합하는 과정은 각각의 원시스

키마로부터 시작된다. 원시스키마들은 통합전 CDM이라 불리우는 같은 모델로 표현되어 있어야 한다. 이것은 만일 원시스키마가 공통의 데이터 모델과 다른 데이터 모델로 표현되어 있다면 스키마 변환을 해야 함을 의미한다. 스키마들은 정확하고 완전해야 한다. 스키마의 정확성과 완전성을 얻기 위해서 다음과 같은 기법들을 사용한다.

- 모델링 안내지침을 따른다.
- 모델의 무결성 제약조건을 시행한다
- 사전정보를 기록한다
- 공통교환 양식을 사용한다.

단계 2 : 스키마 분석

스키마분석 단계에서는 같은 현실 세계를 나타내는 서로 다른 스키마 객체들이 합병될 수 있도록 원시스키마들이 분석되어진다. 효과적인 스키마 분석을 위하여 스키마 브라우징, zooming과 같은 그래피칼 디스플레이 기능, 그래피칼 질의어 능력 등이 요구된다. 이 단계에서는 이질 원시스키마간의 충돌이 발견되고 해결되어야 한다.

단계 3 : 통합

이 단계에서는 확장된 ER모델에서의 스키마통합에 촛점을 맞추어 여러 스키마를 한꺼번에 통합하는 것보다는 우선 한번에 두 스키마씩만 통합한다. 통합하는 순서는 우선 attribute assertion을 결정하고, 객체(엔티티)를 통합한 다음, 릴레이션쉽의 통합을 수행한다. 이때 릴레이션쉽의 통합시에는 릴레이션쉽의 degree, 역할(role), 제약조건 등을 고려하여 합병 작업을 실시해야 한다.

단계 4 : 재구성 및 문서화

통합후에라도 부수적인 시맨틱을 반영하기 위해서 통합된 스키마를 재구성할 수도 있다. 통합 결과를 분석하는 능력은 매우 중요하다. 문서화란 사전 정보와 스키마에 대한 주석을 달고 보고서를 작성하는 작업을 말한다.

(2) Batini et al의 통합 기법 [Bat 86]

단계 1: : Preintegration

이 단계에서는 통합에 참여할 스키마들이 선택되고 통합의 순서, 통합에 적용될 특별사항 등이 분석되어진다. 또한 통합에 사용될 부수적인 정보, 예를 들면 뷰간의 제약조건 등도 분석되어진다.

단계 2 : Comparison

이 단계에서는 개념들간의 대응을 결정짓기 위해 스키마들이 비교되고 충돌들이 분석되어진다. 스키마 비교작업 도중에는 스키마간의 성질이 발견되어진다.

단계 3 : Conforming

스키마 비교시 발견된 충돌을 해결하기 위한 노력이 진행된다. 이 충돌이 해결되어야 스키마의 합병이 가능해지기 때문이다. 충돌 자동 해결은 일반적으로 가능하지 않다. 효과적인 충돌해결을 위해서는 실제 통합 작업에서는 DB 설계자와의 지속적인 접촉이 필요하다.

단계 4 : Merging and Restructuring

스키마들이 합병되어 새로운 스키마를 만들어 낸다. 생성된 스키마는 다시 분석되어 요구수준에 미달되는등 필요하다면 재구성될 수도 있다. 이때 통합된 전역스키마는 다음과 같은 기준에 의해 평가되어진다.

- Completeness and Correctness: 통합된 스키마는 통합에 참여한 구성스키마에 표현된 모든 개념을 포함해야만 한다. 또한 통합된 스키마는 구성스키마들과 연관된 모든 응용의 합(union)이어야 한다.
- Minimality: 같은 개념이 하나 이상의 구성스

키마에 표현되었다면 통합스키마에서는 이 개념이 반드시 한번만 표현되어야 한다.

- Understandability: 통합스키마는 설계자와 사용자 모두에게 이해하기 쉬워야 한다. 이는 즉 통합의 결과를 표현하는 여러가지 방법중에 가장 이해하기 쉬운 형태로 표현되어야 함을 의미한다.

232 스키마통합 순서

스키마통합 순서는 통합되어질 스키마들을 어떤 순서로 통합할것인가를 다룬다. 스키마통합 순서를 결정하는 일은 preintegration 단계에서 수행되는 작업으로서 이는 스키마통합 작업의 전체 성능에 중요한 영향을 미치는 요소이다. 일반적으로 각 단계에서 통합에 고려되는 스키마의 수는 n≥2가 될 수 있다. 다음 그림 27은 통합 순서에 관한 기법 4가지를 보여주고 있다. 각 기법은 모두 트리의 형태로 표현된다. 리프 (leaf) 노드는 통합에 참여한 구성스키마를 나타내고, 넌리프(non-leaf) 노드는 통합의 중간 결과를 나타낸다. 루트(root) 노드는 최종 통합 결과를 나타낸다.

통합 기법은 그림 27에서 보는 바와 같이 우선 binary와 n-ary로 나뉜다.

이 중 binary 기법은 한번에 두 스키마씩 통합하는 방법으로 이는 다시 ladder 기법과 balanced 기법으로 나뉜다. Ladder 기법은 새로운 구성스키마가 이미 존재하는 중간결과에 통합되는 기법이고, balanced 기법은 구성스키마들을 처음부터 한쌍씩 통합하고 이 결과를 다시 통합해나가는 기법이다. 한편 n-ary 기법은 한번에 n개의 스키마를 통합하는 방법으로 이는 one-shot 기법과 iterative 기법으로 나뉜다. One-shot 기법은 n개의 스키마가 모두 한 단계에서 한꺼번에 통합되는 기법이고, iterative 기법은 n개의 스키마가 여러 단계로 나뉘어 통합되는 기법이다.

N-ary 기법의 장점으로서는 상당한 부분의 시맨틱 분석이 합병 이전에 수행될 수 있어 통합 작업의 작업량을 줄일 수 있고 또 통합 절차의 수도 줄일 수 있다. 그러나 n이 커짐에 따라 복잡도가 크게 증가하는 단점이 있다.

Binary 기법의 잇점은 스키마통합시 스키마 분석 및 통합 활동이 간단해진다는 점이다. 현재까지 연구된 스키마통합 기법중 대부분은 통합될 스키마의 수와 관련된 통합의 복잡성 증가때문에 binary 기법을 사용하고 있다. 그러나 binary 기법의 단점은 통합 연산의 수가 증가하고 통합후 빠진 부분이 없나 검사하는 최종분석 작업이 필요하다는 점이다.

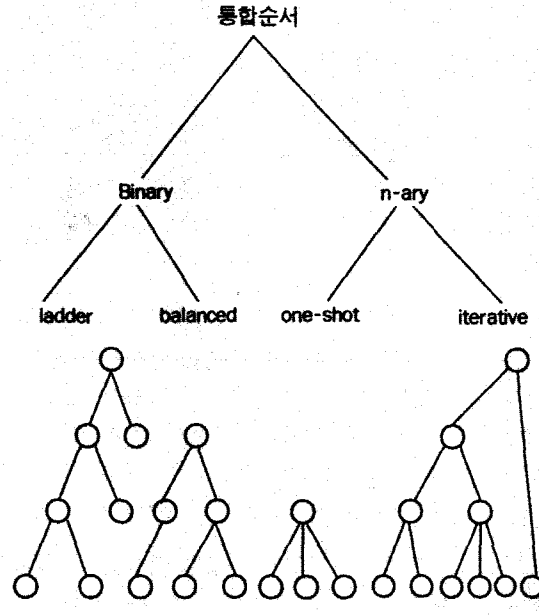


그림 27 스키마통합 순서 기법

233 충돌 분석 및 해결

충돌 분석 및 해결은 어떤 통합절차를 사용해서 통합을 실시하건 간에 반드시 수행되어야할 핵심적인 작업으로 이에 관한 연구는 계속 진행되고 있다. 여기서 충돌분석이란 두 스키마간의 같은 대상을 나타냄에 있어서의 모든 차이를 알

아내는 것을 말한다. 충돌분석에는 크게 나누어 스키마내의 개념의 이름을 비교하여 통일시키는 이름 충돌 분석과, 스키마내의 개념의 표현을 비교하여 통일시키는 구조 충돌 분석이 있다.

(1) 이름 충돌 분석

이름 충돌을 야기하는 두가지 근원은 동음이의어(homonym)와 동의어(synonym)이다.

● 동음이의어 : 응용 도메인의 서로 다른 객체가 두 스키마에서 같은 이름으로 표현되었을 때 발생하며 이 충돌이 발견 해결되지 않으면 불일치를 초래한다. 예를 들어 대학의 학과가 보유하고 있는 '장비'와 빌딩이 소유하고 있는 '장비'는 이름은 같으나 그 의미는 다르다. 즉, 대학에서의 장비는 컴퓨터, 복사기 등을 가리키고, 빌딩에서의 장비는 에어콘 등과 같은 가구류를 가리킨다. 따라서 이러한 두 이름을 가진 스키마를 합병하는 것은 개념적으로 서로 다른 두 객체를 하나의 객체로 만들어내는 결과를 초래한다.

● 동의어 : 응용도메인의 한 객체가 두 스키마에서 서로 다른 이름으로 표현되었을때 발생한다. 만일 서로 다른 이름이 더 나은 이해를 돕지 않는 한 이 충돌은 해결되어야 한다. 예를 들면 '고객'과 '손님'이다. 이 두 단어는 실생활에서 같은 개념을 나타내고 있다.

동의어와 동음이의어를 발견함에 있어 데이터베이스 설계자는 개념간의 불일치 또는 유사성에 의해 도움을 받는다. 개념의 유사성 (concept similarity)는 서로 다른 이름의 두 개념이 같은 성질과 제약조건을 가질때 존재하고, 개념의 불일치는 같은 이름의 개념들이 서로 다른 성질과 제약조건들을 가질때 존재한다. 유사성은 동의어를, 불일치는 동음이의어를 야기한다. 유사 개념 및 불일치 개념을 발견해낸 뒤에는 이름의 재부여와 같은 스키마를 수정하는 과정이 뒤따르게 된다.

(2) 구조 충돌 분석

구조적 충돌은 다음과 같이 분류될 수 있다.

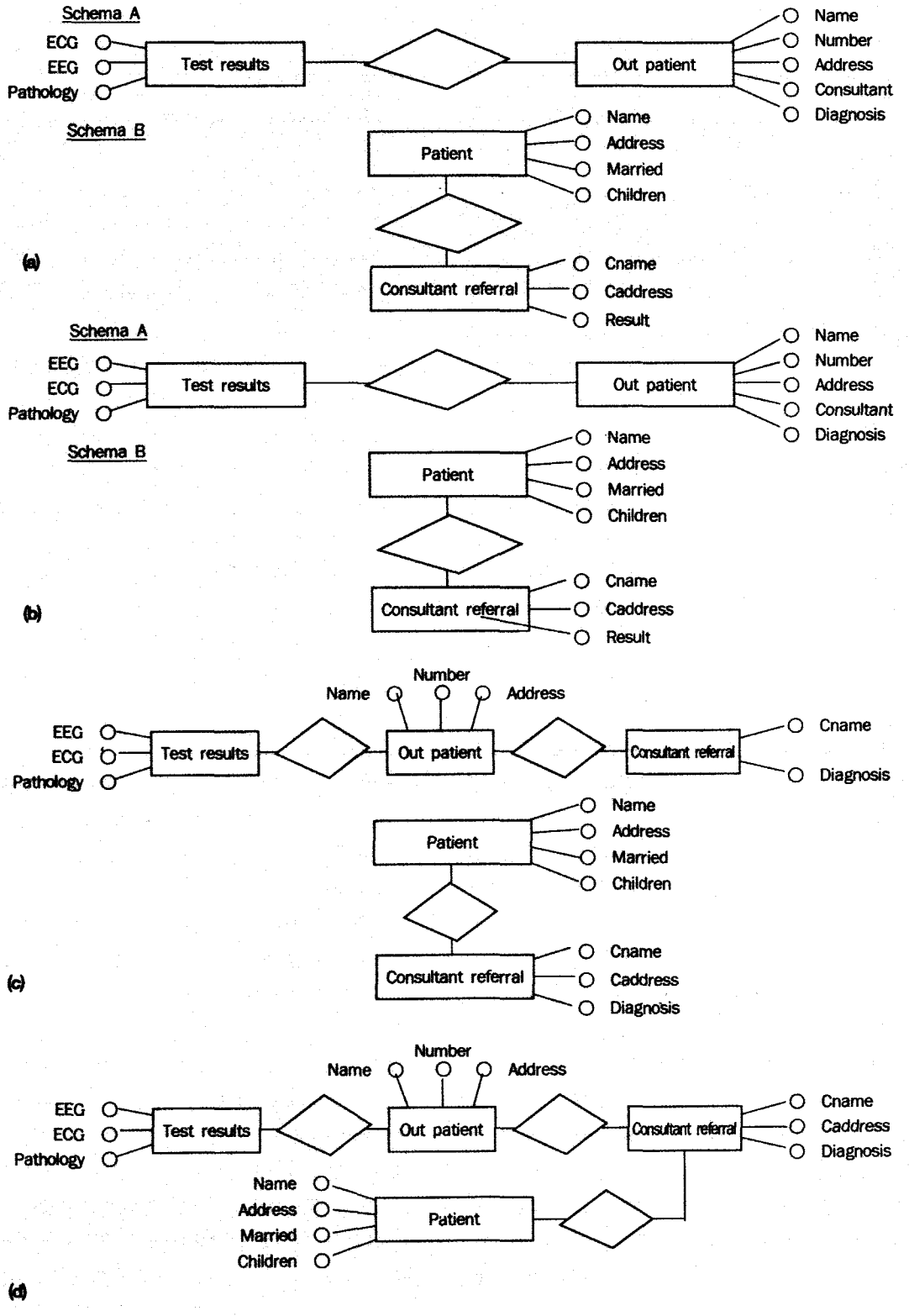
● Type 충돌 : 같은 개념이 다른 스키마에서 다른 구조를 모델링하는데 사용될때 발생한다. 예를 들면 한 스키마에서 엔티티가 다른 스키마에서는 애트리뷰트로 표현될 수 있다.

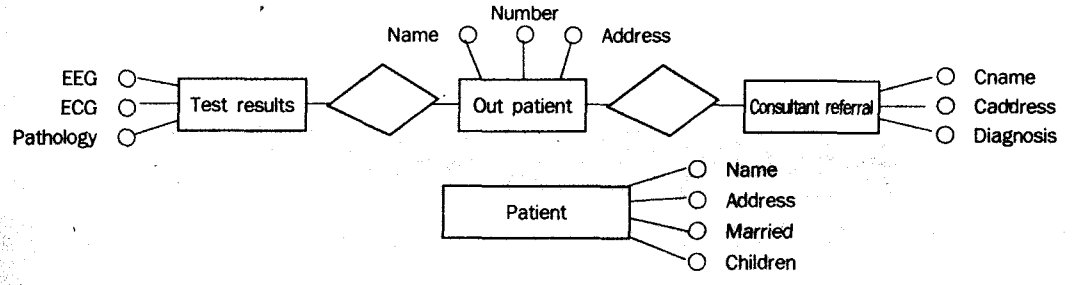
● Dependency 충돌 : 한 그룹의 개념들이 다른 스키마에서 다른 dependency와 연관되어질때 발생한다. 예를 들면 남자와 여자사이의 결혼 관계는 한 스키마에서는 1:1 관계로, 다른 스키마에서는 m:n 관계로 표현될 수 있다.

● Key 충돌 : 다른 키들이 다른 스키마에서 같은 개념을 나타내는데 사용될때 발생한다. 예를 들어 '주민등록번호' 애트리뷰트와 '사번' 애트리뷰트는 서로 다른 두 스키마에서 각각 '회사원' 릴레이션의 키가 될 수 있다.

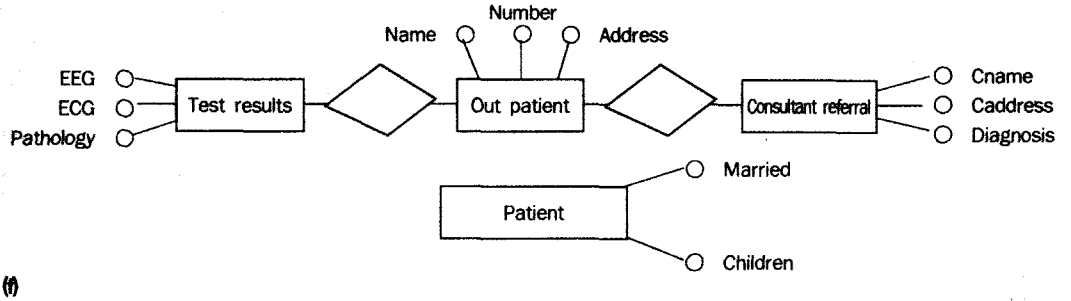
● Behavioral 충돌 : 이 충돌은 다른 삽입/삭제 방침들이 서로 다른 스키마에 있는 같은 객체들과 연관될때 발생한다. 예를 들어, 한 스키마에서는 부서는 회사원 없이도 존재하는 것을 허용할런지도 모르지만 다른 스키마에서는 마지막 회사원을 삭제할때 그 부서 자체까지도 삭제할 수가 있다.

이와 같은 구조적 충돌이 발생했을 경우에는 두 스키마중 하나를 변화시킴으로서 쉽게 해결 가능한 것도 있으나 그렇지 않은 경우는 두 스키마를 충분히 지원하는 공통 표현법에 의해 해결해야 한다. 충돌은 이 외에도 missing data나 도량형 단위의 차이로도 발생할 수 있다. 이러한 충돌은 보조 데이터베이스나 도량형 변환규칙에


Schema A
ECG
EEG
Pathology
Test results
Out patient
Name
Number
Address
Consultant
Diagnosis
Schema B
Patient
Name
Address
Married
Children
Consultant referral
Cname
Caddress
Result
(a)
Schema A
EEG
ECG
Pathology
Test results
Out patient
Name
Number
Address
Consultant
Diagnosis
Schema B
Patient
Name
Address
Married
Children
Consultant referral
Cname
Caddress
Result
(b)
Number
Name
Address
EEG
ECG
Pathology
Test results
Out patient
Consultant referral
Cname
Diagnosis
Patient
Name
Address
Married
Children
Consultant referral
Cname
Caddress
Diagnosis
(c)
Number
Name
Address
EEG
ECG
Pathology
Test results
Out patient
Consultant referral
Cname
Caddress
Diagnosis
Name
Address
Married
Children
Patient
(d)



(e)



(f)

그림 28 스키마 통합 예

의해서 해결될 수 있다.

2.3.4 스키마통합 예

이 절에서는 간단한 예를 들어 스키마 통합이 어떻게 이루어지는가를 보인다.

〈예〉

두 구성스키마 A, B가 있다. 스키마 A는 병원의 외래환자 부서에 대한 스키마이고 스키마 B는 이 환자들에 대한 의사들의 데이터베이스 스키마이다. 그림 2.8(a)는 이 두 스키마를 보여준다. 목표는 이 두 스키마를 합하여 하나의 통합된 전역스키마를 만들어내는 것이다. 먼저 스키마 B에서 'result'는 스키마 A에서의 'diagnosis'의 동의어이므로 결과스키마에서는 이 중 diagnosis로 사용하기로 결정하여 스키마 B에서도 result 대신 diagnosis를 사용한다. 그림 2.8(b)는 이 결과를 보여준다. 다음은 스키마 A에서 'consultant'는 Outpatient 엔티티의 애트리뷰트인 반면 스키마 B에서는 이 consultant가 하나의 엔티티 'Consultant Referral'로 표현되어 있다. 이때 만일 consultant를 전역 레벨에서 하나의 엔티티로 표현하는 것이 더 적합하다면 스키마 A의 consultant를 그림 2.8(c)에서와 같이 엔티티로 해야 한다.

이제 이 두 스키마를 합병해 보자. 먼저 Consultant Referral 엔티티를 두 스키마 간의 링크로 하여 합병하면 그림 2.8(d)와 같은 결과를 얻는다. 다음 Patient 엔티티는 실제로 Outpatient의 서브셋이므로 그림 2.8(e)에서와 같은 서브셋 관계를 생성할 수 있다.

마지막으로 Patient와 Outpatient 엔티티 간에는 공통 특성들이 있으므로 이 공통 특성들을 Patient로부터 떼어내면 Patient에는 꼭 Patient에만 해당되는 특성들만 남아있게 된다. 이와같이 해서 통합된 최종스키마는 그림 2.8(f)에서 보는 바와 같다. DB