의미객체모델을 이용한 데이터베이스 뷰 통합용 설계 지원 시스템

이희석* · 임병학** · 김영삼* · 홍의기*

A Design Support System for Integrating Database Views using Semantic Object Model

Heeseok Lee* · Byunghak Leem** · Youngsam Kim* · Euiki Hong*

ABSTRACT

Integrating database views is an important step in the conceptual database design process. This paper develops a view integration support system by using a semantic object model. In order to determine the order of the integration, affinities among views and objects are analyzed by employing the vector space theory. Semantic conflicts such as naming and structural conflicts are then resolved. The resolution results are stored in a view repository. Objects and views are integrated and stored in this view repository until all views are considered. A prototype for the system is built and can be used in a client/server environment.

1. 서 론

경영 환경은 시대에 따라 많은 변화를 한다. 90년대에서는 데이터베이스(Database) 및 네트웍(Network) 중심의 클라이언트/서버(Client/Server), 분산 시스템(Distributed System), 다운사이징(Downsizing), 개방형 시스템(Open System) 등으로 변화 발전하고 있다.

이 중에서 데이터베이스 분야는 파일 시스템(File System)에서 전통적인 데이터베이스로, 또한 그런 전통적인 것에서 관계형(Relational) 데이터베이스, 객체지향(Object-Oriented) 데이터베이스, 멀티미디어(Multimedia) 데이터베이스, 분산 데이터베이스 분야로 발전하고 있다[Date, 1995]. 기업은 이러한 다양한 데이터베이스를 전략적으로 활용함으로써 타 기업에 경쟁 우위를 확보하기 위한 통합된 기업 데이터베이스 인프라(Infrastructure)를 구축하는데 중점을 모으고 있다.

* 한국과학기술원 테크노경영대학원 경영정보전공
** 삼성 데이타 시스템(주)

이런 추세에 맞춰 오늘날 대다수 기업들은 정보 시스템을 구축하거나 이미 구축한 상태에 있다. 그러나 대부분의 정보 시스템들이 개별적으로 구축되어 자원 중복에 의한 낭비가 심화 됨으로 근래에 와서는 시스템 통합(System Integration)이 중요시 되고 있다. 기업의 각 업무별로 구축된 정보 시스템은 각각의 데이터 스키마(Schema)와 데이터베이스를 가지고 있게 된다. 이런 데이터베이스에는 여러 가지 문제점이 내포되는 바, 첫째는 업무 변화에 따른 데이터베이스의 확장성 및 융통성의 부족으로 인한 타 시스템과의 연계가 어렵고, 둘째로 사업 부문별, 적용 업무별로 산재한 데이터베이스들의 정보들에 대한 일치성 및 관리가 어렵다는 것이다. 마지막으로 데이터베이스 관리 시스템(Database Management System)간의 호환성 부족으로 올바른 시스템 통합이 어렵다는 점이다.

이러한 문제점들을 해소하기 위해서 기업들은 정보 전략 계획(Information Strategy Planning)의 일환으로 기업 통합 데이터 모델을 구축하고 있다[Martin, 1989]. 이러한 통합 데이터 모델을 구축하기 위해서 우선적으로 기업의 단위 업무 영역에서 요구되는 개별 데이터 모델을 구축한 다음, 각각의 개별 데이터 모델을 합병하여 하나의 통합 데이터 모델을 구축하여야 한다. 개별 데이터 모델 구축은 기업 내에 산발적으로 존재하는 데이터, 즉 입출력 양식(I/O Form), 보고서(Report), 기존 정보 시스템 내의 데이터 구조, 현업 사용자의 지식 등으로부터 사업 부문별, 적용 업무별 개념적 데이터 모델을 만든다. 이런 데이터 모델을 구축함에 있어 처음부터 기업 통합 데이터 모델을 만들 수도 있지만 개개의 단위별로 따로 개발할 수도 있다. 이 두 가지 경우 모두 데이터 모델 개발자 간의 대화 상의 문제점으로 인해 다음과 같은 문제점이 발생할 수 있게 된다. 첫째, 다른 데이터 모델 간에 동일한 데이터를 중복하여 표현할 수 있다. 둘째, 동일한 데이터에 서로 다른 이름을 명명할 수 있다. 셋째, 다른 데이터에 대해 동일 이름을 명명할 수 있다. 이상의 문제점들은 데이터의 올바른 관리를 어렵게 한다. 또한 정보 시스템의 성공을 저해하는 요인이라 할 수 있겠다. 이상의 문제점들을 해결하고, 각 모델을 합병하는 작업을 거쳐 하나의 기업 데이터 모델로 구축하는 과정이 뷰 통합(View Integration)이라 할 수 있다. 상기한 것처럼 뷰 통합 과정은 기업 통합 데이터 모델의 구축에 있어 중요한 과정이다.

본 논문에서는 의미객체모델(Semantic Object Model)을 이용하여 새로운 뷰 통합 방법이 제안되었다. 이에 관한 기본 구조는 본 통합 방법의 뷰 친밀도(Affinity) 분석을 통하여 체계적인 통합 전략을 제공한다. 나아가서 본 통합 방법에 근거한 뷰 통합 지원 시스템이 구축되었다.

본 논문의 구성은 하기와 같다. 2장에서는 개체-관계모델(Entity-Relationship Model)을 이용한 뷰 통합에 대한 문헌 연구를 통하여 본 논문에서 개발된 방법론과 비교한다. 3장에서는 의미객체모델을 이용한 뷰 통합 방법론을 제안하며, 4장에서는 본 통합 시스템의 아키텍처(Architecture)가 기술된다. 마지막으로 본 논문의 5장에서는 기존 연구와의 차이점 및 향후 연구 방향의 결언으로 마치게 된다.

2. 관련 연구

뷰 통합 방법론에 대한 연구는 하기와 같이 진행되었다. 본 장에서는 기존의 연구를 검토하여 본 연구에 활용하기로 하겠다.

Batini, Lenzerini & Navathe[1986]는 개념적 설계 단계에서 개체-관계모델을 이용한 뷰 통합과

스키마 통합 방법론을 제시하고 있다. 통합 절차는 사전 통합(Preintegration), 뷰 비교(Comparison), 뷰 확인(Confirming), 그리고 합병 및 재구조화(Merging and Restructuring)를 통해 수행된다. 사전 통합에서는 통합해야 할 뷰의 선택과 통합 전략 결정을, 뷰 비교 단계에서는 이름 충돌(Naming Conflict)과 구조적 충돌(Structural Conflict), 그리고 스키마 간 특성을 찾아 낸다. 이름 충돌은 개체 이름 충돌, 즉, 이음동의어(Synonym)와 동음이의어(Homonym)를, 구조적 충돌은 동일 개념을 서로 다른 구조로 사용한 타입 충돌(Type Conflict), 동일 개념에 대해 수제한(Cardinality) 표기를 서로 다르게 한 종속 충돌(Dependency Conflict), 동일 개념에 서로 다른 식별자를 표현한 키 충돌(Key Conflict), 동일 개념에 서로 다른 삽입/삭제를 수행하는 행위 충돌(Behavioral Conflict)을 찾아 낸다. 이 단계에서 스키마 간 특성(Interschema Properties)도 찾아 낸다. 뷰 적합 단계는 통합을 위해 비교 단계에서 발견된 충돌들을 해소한다. 또한 합병이 용이하도록 스키마 변환을 한다. 마지막으로 합병 및 재구조화 단계는 공통 개념의 통합화(Superimposition)를 이루고, 완전성, 최소성, 이해성을 높일 수 있도록 스키마들의 재구조화를 다룬다.

Navathe, Elmasri & Larson[1986]은 개체-관계모델의 확장인 개체-범주-관계모델(Entity Domain Relationship Model)을 기반으로 통합 방법을 다루고 있으며, 통합 방법은 한번에 두개의 스키마를 통합하는 이진(Binary) 방법을 이용하고 있다. 여기서 범주의 개념으로는 일반화 계층과 관계상에서의 개체가 어떻게 관계에 참여하는지를 명시하는 구조적 제약 개념을 개체-관계모델에 첨가하여 확장한 개념이다. 통합 절차는 사전 통합, 유사 개체의 통합, 서로 다른 개체의 통합, 관계 통합을 거쳐 수행된다. 사전 통합 단계에서 개체와 속성 이름의 일관성을 분석하고, 각 개체의 후보키와 도메인을 정의한다. 그리고 동일 속성을 갖는 개체들 간의 사상(Mapping)을 명시한다. 이 단계에서 이름 및 구조적 충돌의 해결은 완료되었음을 가정하고 개체와 관계의 통합 방법을 구사하고 있다. 유사 개체의 통합 단계는 개체 인스턴스들의 집합 개념인 도메인을 이용하여 동일 도메인, 포함 도메인, 겹침 도메인, 배타적 도메인으로 개체를 분류하여 각각 통합 방법을 제시하고 있다. 서로 다른 개체의 통합은 유사 개체 통합 방법의 일환이다. 마지막으로 관계 통합은 개체의 통합과 동일한 방법으로 이루어진다.

Batini, Ceri & Navathe[1992]는 확장된(Extended) 개체-관계모델을 이용한 통합 방법론을 제시하고 있으며, Batini, Lenzerini & Navathe [1986]의 통합 방법을 한층 발전 시켰다고 할 수 있다. 통합 절차는 충돌 분석 해결과 합병으로 구성된다. 충돌 분석 해결은 이름 충돌과 구조적 충돌 분석으로 구성된다. 이름 충돌은 개념 유사성(Concept Similarity)에 의해 이음동의어를 찾아내고, 개념 불일치성(Concept Mismatch)에 의해 동음이의어를 찾아 내어 이름 변경(Renaming)을 통해 충돌을 해결한다. 또 구조적 충돌은 동일 개념, 호환 개념, 비호환 개념(서로 다른 수제한, 서로 다른 식별자)에 대해 다룬다. 마지막으로 공통 개념을 통합하는 합병을 수행한다. 완전히 일치하는 개체는 어느 한 개체를 삭제하고, 동일한 실세계를 나타내고 있지만 서로 다른 속성을 가지고 있는 개체의 경우는 속성들을 합쳐서 하나의 개체로 통합하며, 스키마 간 특성이 있을 경우는 일반화 개념을 이용하여 뷰 간 개체 구조를 변환하고 통합을 수행한다.

Shin et al.[1994]은 개체-관계모델로 작성된 데이터 모델을 대상으로 작성된 개별 뷰를 하나의 개념 뷰로 통합하는 뷰 통합 절차와 통합된

뷰에서 중복 표현된 개념을 찾아내어 처리하는 방법을 다루고 있다. 뷰 통합 절차는 뷰 통합 전처리 과정, 개체들의 통합, 관계들의 통합, 중복 관계의 처리로 이루어진다. 뷰 통합 전처리 과정은 통합 대상이 되는 뷰들을 선정하고, 이름이나 구조상의 충돌을 해결한다. 개체들의 통합은 개체들의 도메인을 분석하여 동일(Identical), 포함(Contained), 겹침(Overlapping), 겹치지 않는(Disjoint)개체의 형태로 구분하고 이들간의 통합 방안을 강구한다. 관계들의 통합은 개체들이 통합된 후 이들에 부속되어 있는 관계들 간의 관계를 동일, 수제한만 다름, 부분집합(Subset), 겹침, 겹치지 않는 관계의 형태로 분류하고, 통합된 관계들과 기존의 관계들을 함께 고려하여 중복되어 표현된 것들을 발견하고 이것의 처리 방안을 제시하였다. 중복 표현된 관계를 발견하기 위하여, 관계들의 수제한을 이용하고, 개체-관계모델을 유방향 그래프로 변환하여, 이를 이용한 중복 관계를 찾아내어 해결하는 방안을 제시하였다.

3. 의미객체모델을 이용한 뷰 통합 방법론

3.1 의미객체모델(Semantic Object Model)

의미객체모델은 사용자의 요구 사항을 도출하는 개념적 데이터 설계를 위한 데이터 모델 중의 하나이다. 개체-관계모델[Chen, 1977]은 데이터 중심으로 실세계 표현하는 기법이다. 객체모델[Booch, 1994]은 모든 것을 객체로 보고 그 객체에 데이터와 프로세스를 함축시켜 실세계를 표현하는 기법이다. 의미데이터모델[Hull & King, 1987]은 개체-관계모델과 마찬가지로 데이터 중심의 분석 기법인데 의미를 보다 많이 표현해줄 수 있는 기법이다. 의미객체모델과 타 데이터 모델을 비교하면 하기와 같다.

의미객체모델은 개체-관계모델과 여러 가지 면에서 유사성을 가지고 있지만 몇 가지 차이점을 기술하면 다음과 같다. 첫째, 의미객체모델에는 관계의 표현이 속성으로 나타내어진다. 이것은 개체-관계모델에서는 정보들의 연결성을 관계를 통해서 표현하지만 의미객체모델에서는 하나의 의미객체 안에 얼마든지 많은 다른 의미객체를 포함하여 표현함으로 개체-관계모델보다 의미론적으로 완벽하게 표현할 수 있게 된다. 둘째, 기업 양식으로부터 요구 사항을 추출하는데 상대적으로 용이한 방법으로 알려져 있다. 의미객체모델의 의미객체가 기업 양식과 유사한 형태를 취하고 있어서 이해하기가 쉬운 장점을 가지고 있다. 마지막으로 개체-관계모델보다 많은 의미를 표현해주는 기호들을 가지고 있다. 예를 들어 객체와 속성(Attribute) 간의 수제한을 표현할 수 있는 것과 공식 속성(Formula Attribute), 의미객체 연결 속성(Semantic Object Link Attribute) 등을 말할 수 있다[Dewitz and Olson, 1994]. 객체모델은 모든 것을 객체로 표현한다는 점에서 의미객체모델과 같지만 의미객체모델에서는 프로세스를 갖지는 않는다. 또한 의미데이터모델은 실세계를 보다 잘 표현하기 위해 의미론적으로 다른 분석 기법에 비해 보다 많은 기호를 사용한다는 점에서 의미객체모델과 유사하다고 할 수 있다. 하지만 객체로 표현한다는 점에서는 차이점을 보인다.

의미객체모델은 객체를 명확히 구분해 주는 속성들의 집합인 의미객체, 의미객체의 특성을 나타내는 속성, 의미객체를 유일하게 구별할 수 있는 하나 또는 그 이상의 속성인 객체 식별자(Object Identifier), 한 속성이 가질 수 있는 값의 영역을 나타내는 속성 도메인(Domain), 한 객체에 대한 속성들의 인스턴스(Instance)의 수를 나타내는

수제한 등으로 구성된다. 각각의 개념들에 대한 설명이 〈표 3-1〉에 기술되어 있다. 의미객체모델은 Kroenke[1995]에 상세하게 기술되어 있다.

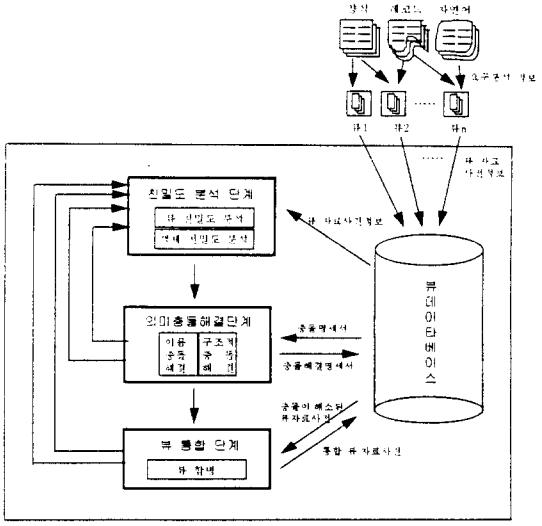
〈표 3-1〉 의미객체모델의 개념[Dewitz and Olson, 1994]

개 념	정 의	예
의미객체	객체를 명확하게 구분해 주는 사람, 장소, 물건 등, 관리가치가 있는 데이터의 집합.	CUSTOMER, SALES STORE_LOCATION, ITEM
속성	의미객체의 특성의 표현으로 단순 속성, 그룹 속성, 공식 속성, 의미 객체 연결 속성이 있고, 인스턴스를 표시할 수 있는 속성의 최소/최대 수제한을 갖는다.	CustomerID(단순 속성) CustomerName(그룹 속성) CUSTOMER(의미 객체 연결 속성)
객체 식별자	객체를 식별할 수 있는 하나 또는 그 이상의 속성.	CustomerID(유일) CustomerName(중복 허용)
속성 도메인	속성의 물리적 특성과 의미 표현. 물리적 특성은 데이터 타입, 크기, 양식, 값의 제약을 포함하고, 의미적 표현은 속성이 가지는 기능이나 응용 프로그램을 말한다.	SalesDate={Month, Day, Year} Month : Integer, 2 Values : {01 to 12}

3.2 뷰 통합 방법론

상기한 의미객체를 이용하여 본 논문에서 제안된 뷰 통합 절차는 3단계로 구성되었다. 첫 단계는 뷰 간의 친밀도를 분석하는 친밀도 분석 단계이고 두 번째 단계는 충돌의 발견 및 해소를 하는 의미 충돌 해결 단계, 마지막 단계는 뷰를 합병하는 뷰 통합 단계이다. 이 절차를 〈그림 3-1〉에 보여주고 있다.

1단계, 친밀도 분석 단계는 의미객체모델에 의해 모델링된 기업의 양식을 뷰 자료 사전 정보를 이용하여 뷰 데이터베이스로 구축하고, 이 데이터베이스로부터 뷰와 객체의 친밀도 분석을 통한 친밀도 값과 통합 순서를 얻게 된다. 따라서 친밀도 분석 단계의 목적은 의미 충돌 분석 및 통합을 위한 순서를 결정하는데 있다. 2단계의 의미 충돌 해결 단계는 1단계에서 결정된 순서에 의해 뷰 간의 의미객체, 속성, 수제한 등의 충돌을 해소하며, 이때 뷰 데이터베이스는 갱신 된다. 3단



〈그림 3-1〉 뷰 통합 절차

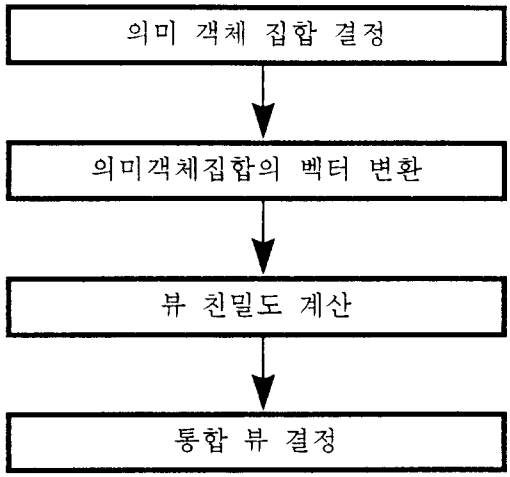
계인 뷰 통합 단계는 2단계의 결과를 입력으로 하여 두 뷰의 부분 통합을 수행하며 마지막 뷰가 모두 합병될 때까지 3단계를 반복 수행한다.

3.2.1 친밀도 분석 단계

친밀도 분석은 휴리스틱(Heuristic) 방법에 의한 뷰 통합 순서를 결정해 주어, 통합을 용이하게 하는데 그 목적이 있다. 친밀도 분석은 두 단계로 이루어진다. 첫번째 단계는 뷰 간의 친밀도 분석이고, 두번째 단계는 결정된 두 뷰의 동일 의미객체 간의 친밀도 분석이다. 이 분석이 완료된 후 친밀도가 가장 높은 뷰들을 모아서 먼저 의미 충돌 해결을 하고 부분 통합을 수행한 다음, 이 부분 통합 뷰와 연결된 뷰의 친밀도를 다시 계산하여 가장 높은 친밀도를 갖는 뷰와 의미 충돌 해결 통합을 수행한다. 모든 뷰가 통합 될 때까지 이 과정을 반복하여 수행한다.

3.2.1.1 뷰 친밀도 분석 단계

첫 단계인 뷰 친밀도 분석 단계의 목적은 통합하고자 하는 모든 뷰 간의 친밀도를 계산하여, 뷰 간의 의미 충돌 해결 및 통합 순서를 결정할 수 있는 기본 자료를 구하는데 있다. 뷰 친밀도란 두 뷰 간에 공통된 정보가 얼마나 많은 가를 나타내는 수치이다. 뷰 친밀도를 계산하기 위해서는 뷰를 의미객체들의 집합으로 보고, 선형 대수의 벡터 이론을 도입하여 계산할 수 있다[Bazaraa & Jarvis, 1977]. 뷰 통합 순서의 결정은 뷰 간의 친밀도가 가장 높은 값을 갖는 것으로부터 뷰 비교 및 통합을 하는 전략을 수립한다. 뷰 친밀도 분석 절차는 의미객체 집합 결정, 의미객체 집합의 벡터 변환, 친밀도 계산, 통합 뷰 결정 순으로 이루어 진다. 이 과정이 〈그림 3-2〉에 도식화 되어 있다.



〈그림 3-2〉 뷰 친밀도 분석 순서

뷰는 단위 업무 영역을 의미객체모델에 의해서 표현한 데이터 모델이다. 이 데이터 모델은 의미객체들로 구성된다. 따라서 뷰는 의미객체들이 모인 집합(Set)으로 볼 수 있다. 의미객체를 O_i (i=1,2,3,…,n), 전체 의미객체들의 집합을 S_o라고 할 때, 다음과 같이 집합으로 표현할 수 있다.

$$S_o=\{O_1,O_2,O_3,\cdots,O_n\}$$

또 뷰가 m개 이고, 각 뷰는 p개 이내의 의미객체를 갖고 있는 경우, 각 뷰를 의미객체 집합으로 표현하면 다음과 같다.

$$S_{oi}=\{O_{i_1},O_{i_2},O_{i_3},\cdots,O_{i_p}\},i=1,2,3,\cdots,m$$

뷰는 의미객체 집합으로 표현되므로 스칼라(Scalar) 값을 가지는 벡터(Vector)로 표현할 수 있다. 통합하고자 하는 뷰가 m개 이고, 그 뷰에 속해 있는 의미객체가 n개일 때, 전체 의미객체 집합을 각 뷰에 대해 벡터로 표현할 경우, 뷰 벡터를 V_i, 의미객체의 스칼라 값을 a_{ik}(k=1,2,3,…,n)라 할 때, 의미객체 집합의 벡터 변환은 다음과 같이 표현될 수 있다.

$$V_i=\{a_{i1},a_{i2},\cdots,a_{in}\},i=1,2,3,\cdots,m$$

$$a_{ik}=\begin{cases}1, O_k\in S_{Oi}\\ 0, O_k\notin S_{Oi}\end{cases},k=1,2,3,\cdots,n$$

즉, 스칼라값 a_{ik}는 뷰 벡터 V_i에 있는 의미객체 O_k의 값이다. a_{ik}의 값은 의미객체 O_k가 뷰 S_{Oi}에 존재할 때는 1의 값을, 존재하지 않을 때는 0의 값을 가진다.

친밀도는 뷰 간에 동일 의미객체가 어느 정도 포함되어 있는가를 나타내는 것으로 스칼라곱(Scalar Multiplication)의 내적(Inner Product)을 이용하여 두 뷰 간의 친밀도 값을 구한다. 친밀도 값이 크면 클수록 뷰 간 동일 의미객체의 수가 많음을 의미한다. 두 뷰 벡터를 각각 V_i,V_j 두 뷰 간의 친밀도를 A_{ij}라고 할 때, 친밀도 값을 계산하는 식은 다음과 같다.

$$A_{ij}=V_i\cdot V_j=\sum_{k=1}^{n} a_{ik}a_{jk}$$

위에서 계산된 친밀도 값에 따라 높은 값을 갖는 뷰의 쌍에 대해서 먼저 의미 충돌을 해결하고 통합을 하여 1부분 통합 뷰를 생성하게 된다. 이 통합 뷰와 나머지 뷰들과의 친밀도 값을 다시 계산하여 그 값이 가장 높은 값을 가지는 뷰의 쌍에 대해서 의미 충돌 해결과 통합을 하게 된다. 이와 같은 과정을 모든 뷰가 통합될 때까지 반복 수행하여 최종 통합 뷰를 얻는다.

앞의 친밀도 계산 과정을 예를 들면 다음과 같다. 의미객체 전체 집합과 각 뷰에 대한 의미객체 집합을 다음과 같이 각각 나타낼 수 있다고 하자.

S_O={NavalOfficer, ActiveDutyMilitaryMember, NavalServiceMember, UNIT, Language, Education, CommissionedOfficer, Security Requirement, YearGroup, Member, MemberOfficer}

S_{O1}={ActiveDutyMilitaryMember, NavalOfficer, UNIT, NavalServiceMember, Language}

S_{O2}={Member, Education, MemberOfficer, SecurityRequirement, Language}

S_{O3}={CommissionedOfficer, Education, YearGroup, UNIT, SecurityRequirement, Language}

이상의 각 뷰에 대한 벡터 변환은 다음과 같다.

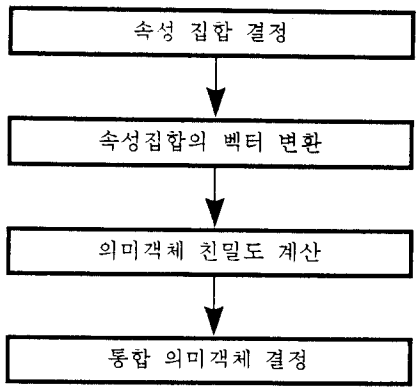
$$V_1=(1,1,1,1,1,0,0,0,0,0,0)$$
$$V_2=(0,0,0,0,1,1,0,1,0,1,1)$$
$$V_3=(0,0,0,1,1,1,1,1,1,0,0)$$

앞서 제시한 친밀도 값 계산식에 의해 계산된 값은 다음과 같다.

$$A_{12}=1,\ A_{13}=2,\ A_{23}=3$$

3.2.1.2 의미객체 친밀도 분석 단계

이 단계에서는 두 뷰 간에 존재하는 의미객체 간의 비교 순서 및 통합 순서를 결정하게 된다. 의미객체 간 친밀도 계산은 뷰 친밀도 계산과 유사한 방법에 의해서 구할 수 있다. 의미객체 친밀도 분석 순서는 뷰 간 속성 집합 결정, 속성 집합의 벡터 변환, 친밀도 계산, 통합 의미객체 결정 순서를 통해 이루어지게 된다. 그 순서는 〈그림 3-3〉에 기술하였다.



〈그림 3-3〉 의미객체 친밀도 분석 순서

뷰가 의미객체들의 집합으로 표현 가능하듯이 의미객체는 속성들의 집합으로 표현이 가능하다. 전 단계의 뷰 친밀도 분석을 통해 선정된 두 뷰(편의상 V_i, V_j라 칭함)의 전체 속성들의 수가 m_a일 때, 전체 속성 집합 S_a는 다음과 같이 표현할 수 있다.

$$S_a=\{att_1, att_2, att_3, \cdots, att_{m_a}\}$$

각각의 뷰 벡터 $V_i(i=1,2,3,\cdots,m)$의 객체 O_k $(k=1,2,3,\cdots,p)$에 속해 있는 속성 집합 S_{aik}는 다음과 같이 표현될 수 있다. 이때 각 객체가 갖는 속성의 수는 최대 n_a개 이다.

$$S_{aik}=\{att_{ik1}, att_{ik2}, att_{ik3}, \cdots, att_{ikn_a}\}$$

각각의 의미객체에 해당하는 속성들의 집합은 스칼라 값을 갖는 벡터로 변환이 가능하다. 뷰 V_i의 객체 O_k에서 속성이 갖는 값들을 a_{ikl}이라 할 때, 속성 벡터 B_{ik}는 다음과 같이 표현할 수 있다.

$$B_{ik}=(a_{ik1}, a_{ik2}, a_{ik3}\cdots, a_{ikm_a})$$

$$a_{ikl}=\begin{cases} 1, att_1 \in S_{aik} \\ 0, att_1 \notin S_{aik} \end{cases}, \quad l=1,2,3,\cdots,m_a$$

즉 a_{ikl}의 값은 전체 속성 집합 S_a에 있는 속성이 각 뷰의 객체 속성 집합 S_{aik}에 속해 있으면 1의 값을, 속해 있지 않으면 0의 값을 가진다.

의미객체 친밀도는 두 의미객체 간에 동일 속성의 보유 정도를 나타낸다. 의미객체 친밀도의 계산은 뷰 친밀도 계산과 같이 두 속성 집합 벡터의 내적에 의해 구할 수 있다. 뷰의 친밀도가 가장 높았던 두 뷰 V_i, V_j에서 뷰 V_i에 있는 객체 이름이 k, 뷰 V_j에 있는 객체 이름이 k', 두 뷰 간에 공통으로 갖는 객체들의 친밀도를 A_{ij}^{a}라 할 때 일반적인 계산식은 다음과 같다.

$$A_{ij}^{a}=B_{ik} \bullet B_{jk'}=\sum_{l=1}^{m_a} a_{ikl}\, a_{jk'l}$$

통합 의미객체의 결정은 객체 친밀도 값이 가장 큰 객체가 우선 의미 충돌 해결 및 통합을 하게 되고 다음 순의 친밀도 값을 갖는 객체가 차례로 충돌 해결 및 통합을 계속한다.

3.2.2 의미 충돌 해결 단계

다음 단계는 의미 충돌 해결 단계이다. 이 단계에서는 친밀도 분석 단계에서 선택된 두 뷰를 의미객체로 표현 시 발생할 수 있는 모든 의미 충돌을 찾아내어 해결한다. 의미 충돌 및 해결을 어느 정도 자동화 시켜 줄 수 있는지의 관건은 뷰 데이터베이스 구성 방법에 달려 있다. 뷰 데이터베이스는 객체명, 속성명, 속성 특성, 수제한으로 이루어져 있다. 이 데이터베이스로부터 구조적 질의어(Structured Query Language)를 사용하여 이름 충돌과 구조적 충돌을 찾아내고 해결한다. 이 단계의 최종 결과는 의미 충돌이 해결된 뷰 데이터베이스 생성이다.

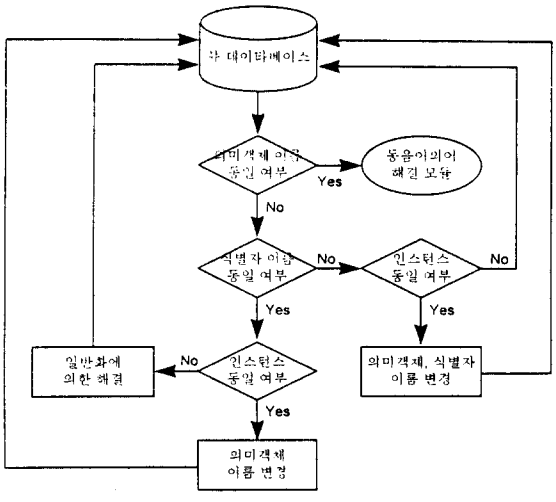
의미 충돌 해결 단계는 이름 충돌 해결과 구조적 충돌 해결로 양분된다. 첫 번째로, 이름 충돌에는 동일한 개념을 다른 이름으로 표현한 이음동의어와, 다른 개념을 동일한 이름으로 표현한 동음이의어가 있다. 이음동의어와 동음이의어는 개념 유사성과 개념 불일치로 분석될 수 있다. 개념 유사성은 서로 다른 이름을 가진 개념이 뷰에서 공통의 특성과 제약을 가질 때 발견된다. 두 뷰에 유사성이 있다라는 것은 이음동의어가 있음을 의미한다. 개념 불일치는 동일 이름을 가진 개념이 뷰에서 서로 다른 특성과 제약을 가질 때 발견된다. 두 개념이 불일치 된다라는 것은 동음이의어가 있음을 의미한다[Batini, 1992; Teory, 1990; Lee et.al., 1995]. 본 논문에서는 객체와 식별자에 의해 이음동의어와 동음이의어를 찾는 방법을 〈표 3-2〉에 제안하였다. 여기서 속성들에

〈표 3-2〉 이음동의어/ 동음이의어 개념과 식별

	이음동의어	동음이의어
개 념	동일 개념을 다른 이름으로 표현	서로 다른 개념을 동일한 이름으로 표현
발 견	뷰 간 식별자와 인스턴스가 동일하고 객체 이름이 다르게 표현된 경우	객체 이름은 동일하지만 식별자와 인스턴스가 다른 경우
해 결	객체 이름 변경, 식별자 이름 변경, 일반화(Generalization)	

대한 의미 충돌 해결은 Y. Kim and H. Lee [1995]가 제안한 CDDSS(Conceptual Database Design Support System)의 방법론을 통해 해결할 수 있다. 그러므로 본 논문에서는 속성들에 대한 의미 충돌은 이미 해결되었다는 가정을 하고 의미객체에 대한 의미 충돌을 해결한다.

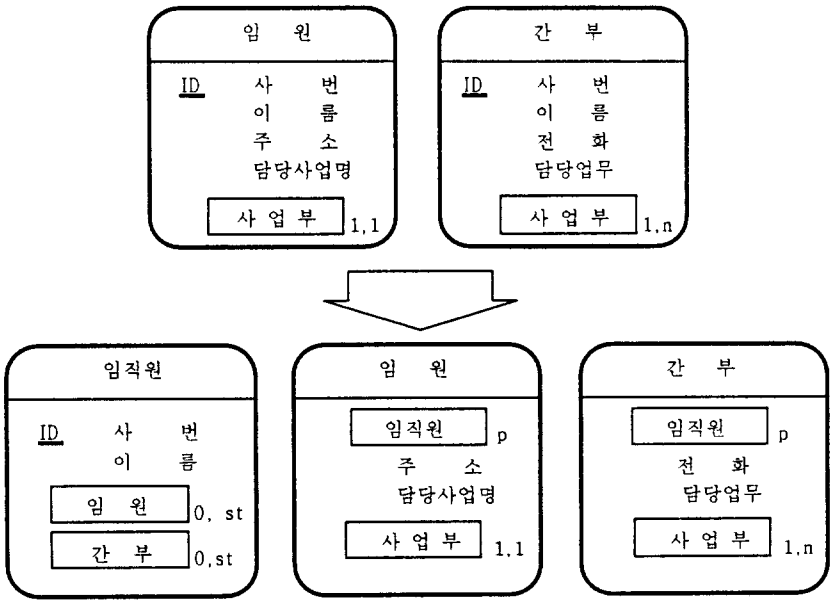
이음동의어는 서로 다른 의미객체간에 동일한 식별자를 갖고 그 인스턴스가 같을 때 발견되고, 인스턴트가 다를 때, 즉 식별자의 인스턴스 간 겹침, 포함, 배타적인 관계에 대한 충돌 문제는 식별자 이름 변경이나 모델의 변환으로 해결할 수 있다. 식별자가 동일한 경우는 식별자에 해당하는 속성의 인스턴스를 비교하여 동일한 경우, 어느 한 쪽의 의미객체 이름으로 변경한다. 식별자가 동일하지 않을 경우는 식별자의 인스턴스를 비교하여 동일한 경우 두 식별자의 이름을 같게하고, 의미객체의 이름도 같게 한다. 이런 변경은 시스템이 자동적으로 해주지는 못한다. 왜냐하면 실제 업무 상의 모든 인스턴스를 데이터베이스로 구축하여 비교할 수 없는 한계 때문이다. 그러므로 인스턴스가 동일하다라는 것은 어느 수준 이상 동일한 인스턴스가 발견되었다는 것이고, 인스턴스가 동일하지 않다라는 것 또한 비교하는 인스턴스에는 동일함이 발견되지 않았을 뿐 실제로 동일하지 않다라는 것은 아니다. 이런 이유로 해서 동일함, 또는 동일하지 않다라는 결과가 나왔을 때, 그 사실 여부는 설계자의 확인을 받아야만 한다. 그러므로 이 과정에서는 뷰 통합 설계자의 주관적 결정이 중요하게 작용한다. 이에 대한 절차를 〈그림 3-4〉에 나타내었다.



〈그림 3-4〉 이음동의어 해결 절차

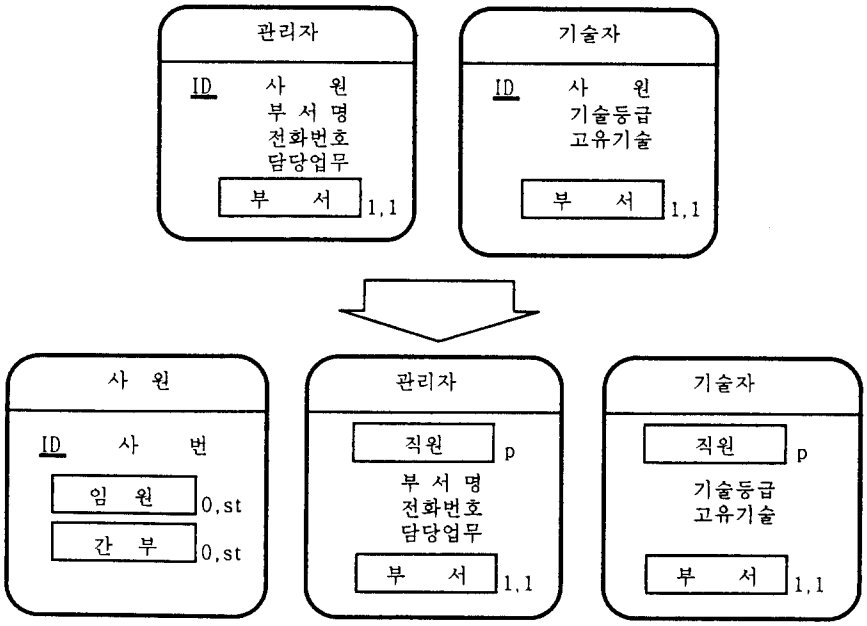
두 의미객체의 식별자 간의 인스턴스가 다른 경우는 상기한 경우 이외에 다음과 같은 경우를 가정할 수 있다[Navathe, 1986;Kamel, 1995;Shin et. al., 1994].

첫번째는 두 의미객체의 식별자가 서로 동일하고 그 식별자들의 인스턴스가 서로 겹치는 경우 즉, 같은 인스턴스도 있고 다른 인스턴스도 있는 경우이다. 이 때는 그 식별자를 갖는 상위(Super) 의미객체를 만들고, 남은 두 의미객체가 그 식별자를 제외한 모든 속성을 포함하도록 하는 하위(Sub) 의미객체를 만들어 준다. 예를 들면, 두 뷰의 의미객체인 "임원"과 "간부"의 식별자는 "사번"으로 동일하고 그 두 식별자의 인스턴스는 서로 겹치는 경우이다. 이런 경우일 때의 충돌 해결은 상위 의미객체로 "임직원"을 만들고 그 속성으로 식별자인 "사번"을 갖도록 한다. 그리고 기존의 의미객체인 "임원"과 "간부"는 의미객체인 "임직원"의 하위 의미객체로 만든 다음 그 속성은 "사번"을 제외한 나머지 속성을 그대로 갖게 하는 것이다. 이에 대한 충돌과 해결 예를 〈그림 3-5〉에 보여 주고 있다.



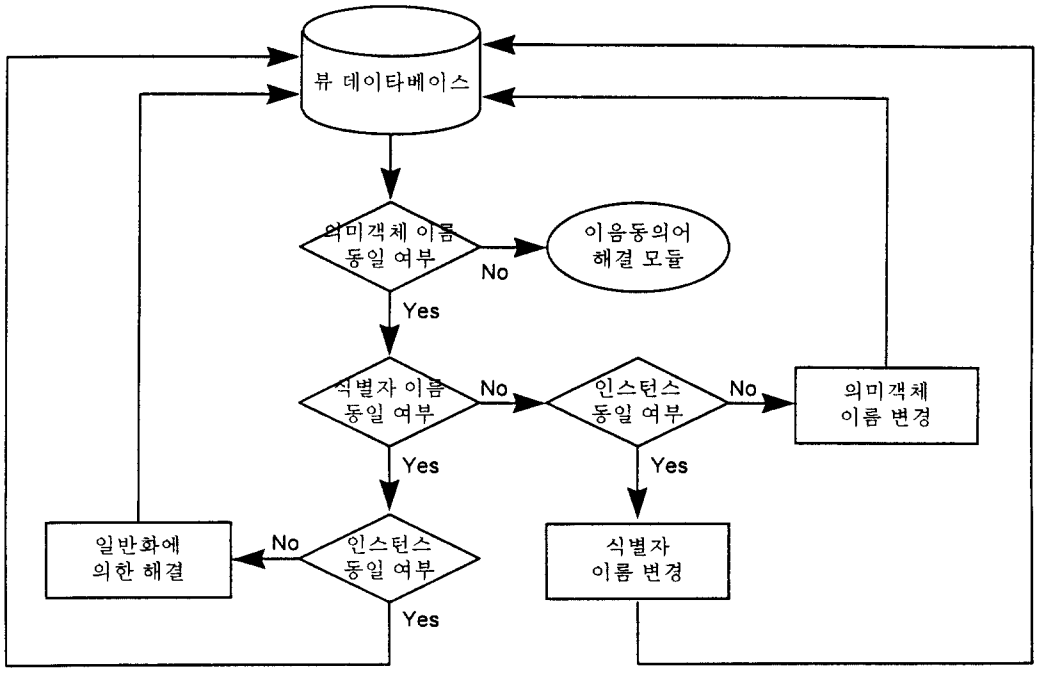
〈그림 3-5〉 식별자 인스턴스의 겹친 충돌 예

두번째는 식별자 이름이 동일하고 인스턴스가 완전히 다른 경우이다. 이 때는 두 의미객체를 모두 포함 할 수 있는 상위 의미객체를 만들고 그 속성은 두 의미객체의 공통 속성을 갖게 한다. 그리고 나머지 속성을 갖는 하위 의미객체를 만든다. 예를 들면 의미객체 "관리자"와 "기술자"인 경우, 식별자가 "사번"으로 동일하고 인스턴스가 완전히 다를 때의 충돌 해결은 상위 의미객체 "사원"을 만들고 기존의 두 의미객체에서 공통의 속성들을 뽑아 상위 의미객체의 속성으로 한다. 그런 다음 기존 의미객체는 하위 의미객체로 만들고 공통의 속성들을 제외한 나머지 속성들을 갖게 한다. 이에 대한 충돌과 해결 예를 〈그림 3-6〉에 보여 주고 있다.



〈그림 3-6〉 식별자 인스턴스가 배타적 관계인 경우의 충돌 해결 예

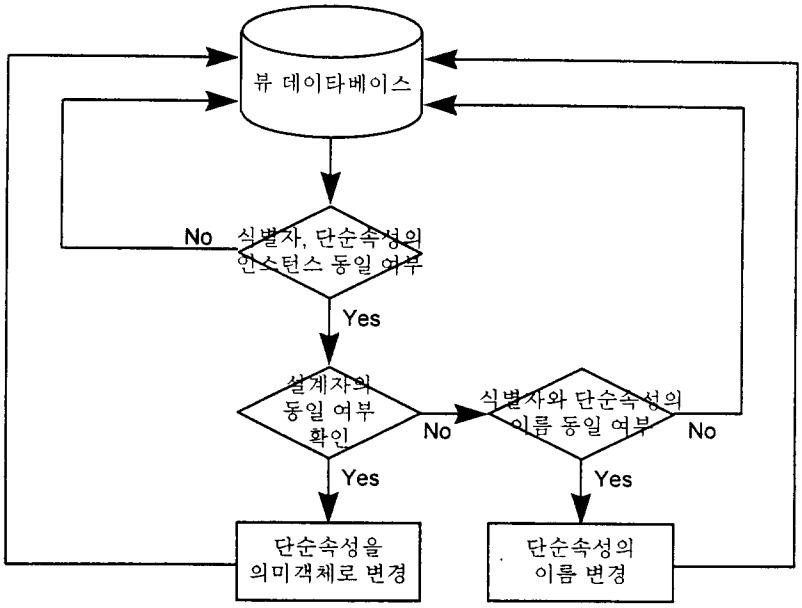
동음이의어 충돌 문제는 인스턴스의 비교-분석을 통해 해결할 수 있다. 이것은 〈그림 3-7〉에 표현되어 있다.



〈그림 3-7〉 동음이의어 해결 절차

의미객체의 이름이 같고 식별자가 동일한 경우에 인스턴스가 다르면 이음동의어 해결과 동일한 방법으로 해결한다. 객체 이름은 같지만 식별자가 동일하지 않을 때, 인스턴스가 동일 할 경우는 어느 한쪽의 식별자 이름을 다른 한쪽의 식별자 이름으로 바꾸어 충돌을 막고, 인스턴스가 다를 경우는 두 의미객체 이름을 다르게 하여 충돌을 해결한다. 이음동의어 충돌 해결 단계와 마찬가지로 이 단계에서도 모든 인스턴스의 비교가 불가능하기 때문에 어느 수준까지 비교하여 설계자의 주관에 따르게 한다.

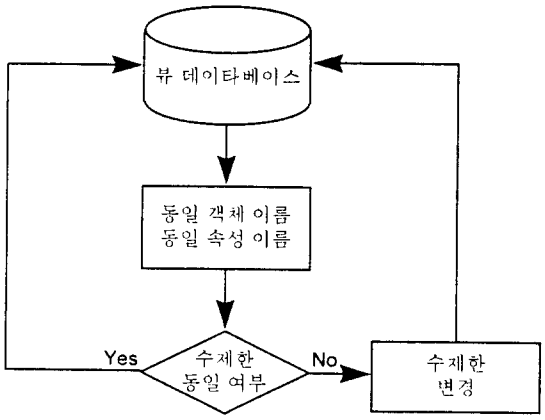
다음 단계는 구조적 충돌 분석 단계이다. 구조적 충돌은 모델링 구조나 제약을 서로 다르게 사용했을 경우에 발생한다. 구조적 충돌에는 타잎 충돌, 수제한 충돌이 있다. 타잎 충돌은 동일 개념이 서로 다른 뷰에서 다른 구조로 표현될 때 일어난다. 예를 들어 동일 개념이 한 뷰에서는 의미객체로, 다른 뷰에서는 단순 속성으로 표현되는 경우이다[Spaccapietra, 1994]. 타잎 충돌의 해결 방법으로는 의미객체의 이름과 속성의 이름과의 동일 여부에 상관없이 의미객체의 식별자의 인스턴스와 단순 속성의 인스턴스가 동일하다면 설계자에게 확인을 받아 동일 개념이라면 단순 속성을 의미객체로 변경하여 해결한다. 이의 절차를 〈그림 3-8〉에 나타내었다.



〈그림 3-8〉 타잎 충돌 해결 절차

객체 이름과 속성 이름에 대한 판단은 통합 설계자가 사용자와의 인터뷰에 의해 판단한다. 변경된 내용에 대해서는 뷰 데이터베이스에 삭제, 추가, 갱신을 한다.

수제한 충돌은 동일 개념을 표현한 속성이나 의미객체 간 수제한이 서로 상이한 경우이다. 의미객체모델은 모든 속성들에 대해서 수제한이 표현되므로 모든 속성들에 대해서 수제한 충돌을 해소해야 한다. 해결절차를 〈그림 3-9〉에 보여주고 있다.



〈그림 3-9〉 수제한 충돌 해결 절차

3.2.3 뷰 통합 단계

뷰 통합 단계는 의미 충돌 해결 단계 완료 후, 두 뷰에서 표현된 모든 개념을 통합 뷰에서 모두 포함되도록 만드는 과정으로 통합된 뷰 데이터베이스를 만들고 통합된 개념 모델을 만드는데 목적이 있다. 뷰 간에 존재하는 이름 및 구조적 충돌을 해결한 후, 뷰 간의 친밀도 분석에 의한 통합 순서에 따라 뷰 간의 통합을 수행한다. 뷰 간 동일 이름을 갖는 객체를 하나의 객체로 만들고 각각의 속성은 하나의 객체에 합친다. 이 과정은 뷰 간 테이블을 통합 순서에 따라 뷰 테이블들을 합(Union)해 주면 된다.

3.3 뷰 통합 방법론 비교

본 논문에서 제안한 방법론(Lee, Leem, Kim, and Hong)과 기존의 방법론을 비교 분석한 결과를 하기에 기술하였다.

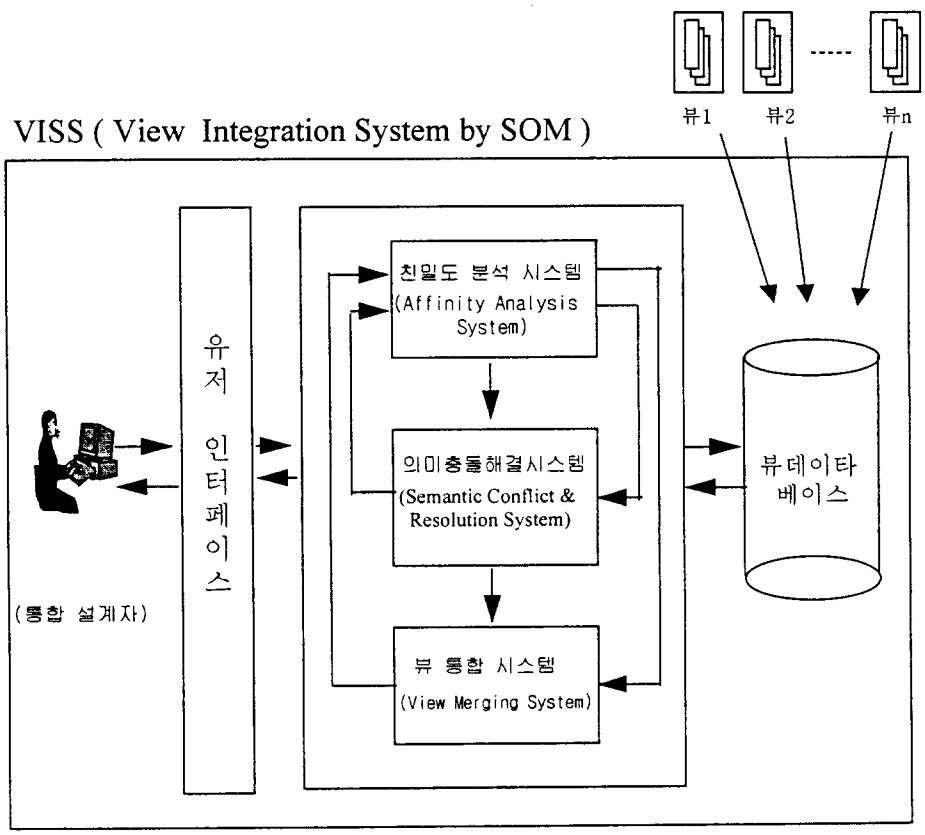
〈표 3-3〉 뷰 통합 방법론 비교

방법론 / 비교 기준	Navathe, Elmasri & Larson (1986)	Batini, Lenzerini & Navathe (1986)	Batini, Ceri & Navathe (1992)	K.SHin, N.Park, J.Park, & J.W. Park (1994)	Lee, Leem, Kim, & Hong (1996)
사용데이타모델	EDRM	ERM	EERM	EERM	SOM
통 합 순 서	Binary	Binary	Binary	Binary	2 Phase Binary
뷰친밀도분석	없음	없음	없음	없음	뷰/객체 친밀도 및 통합 순서 결정
이름충돌해결	기해결 가정	이음동의어 동음이의어	이음동의어 동음이의어	기해결 가정	이음동의어 동음이의어
구조적 충돌	기해결 가정	타잎, 식별자, 카디낼러티	타잎, 식별자, 카디낼러티	기해결 가정	타잎, 식별자, 카디낼러티
통 합 방 식	도메인 정보를 기반으로 한 개체간 합병	합 병	합 병	도메인 정보를 기반으로 한 개체간 합병	식별자의 인스턴스 기반의 관계 설정

4. 통합 지원 시스템 개발

4.1 뷰 통합 지원 시스템 아키텍쳐

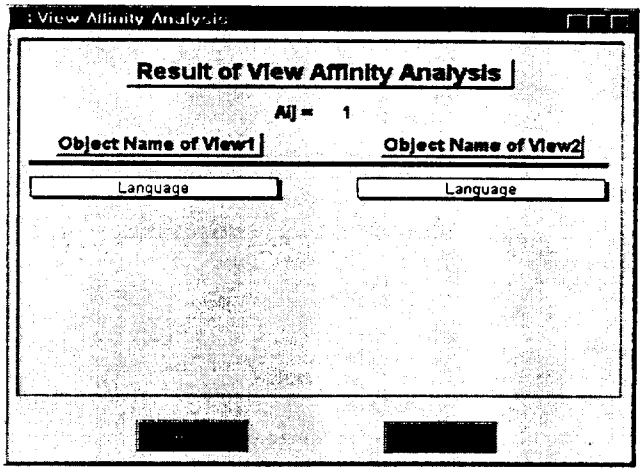
본 장은 제안한 방법론에 근거하여 개발한 뷰 통합 지원 시스템(Views Integration Support System; VISS)의 구성요소에 대해서 설명하고, 이를 기반으로 한 응용 사례를 설명한다. 본 방법론은 통합 준비 단계에서 다양한 뷰에 대한 정보를 데이터베이스화하고, 구축된 데이터베이스로부터 구조적 질의어를 사용하여 뷰 친밀도 분석을 통해 통합될 뷰의 순서를 결정, 그 순서에 의거하여 의미 충돌을 찾아내고 해결할 수 있도록 하였다. 그 다음으로 통합 단계에서는 전 단계에서 결정된 뷰들의 통합 순서에 의해 객체들을 합병하여 하나의 통합 뷰를 생성할 수 있도록 하였다. 시스템 개발 툴은 클라이언트/서버 환경에서 시스템을 개발할 수 있도록 하여 주는 파워빌더(PowerBuilder)를 사용하여 분산된 환경에서도 뷰를 통합할 수 있도록 하였다. VISS는 뷰 통합 설계자로 하여금 주어진 뷰들의 통합을 용이하게 할 수 있도록 지원해 주는 상호 작용적(Interactive)인 시스템이다. VISS의 구성은 〈그림 4-1〉과 같이 사용자 인터페이스, 친밀도 분석 시스템, 의미 충돌 해결 시스템과 뷰 통합 시스템으로 이루어져 있으며 각 시스템에서의 입출력 정보들은 뷰데이터베이스에 저장되어 진다.



〈그림 4-1〉 뷰 통합 지원 시스템 개념도

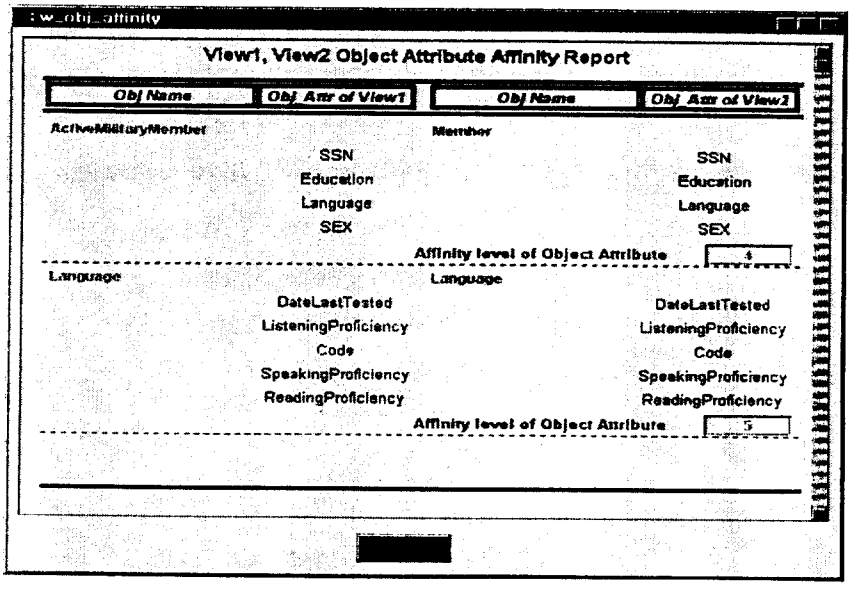
4.2 프로토타입 개발

VISS는 친밀도 분석 시스템, 의미 충돌 해결 시스템, 뷰 통합 시스템의 3개로 구성되어 있다. 우선 뷰 데이터베이스로부터 친밀도를 분석하는 시스템부터 설명한다. 친밀도 분석 시스템에서는 〈그림 4-2〉에서 보는 바와 같이 두 뷰 간에 동일한 이름의 의미객체의 수를 계산하여 그 결과를 보여 준다.



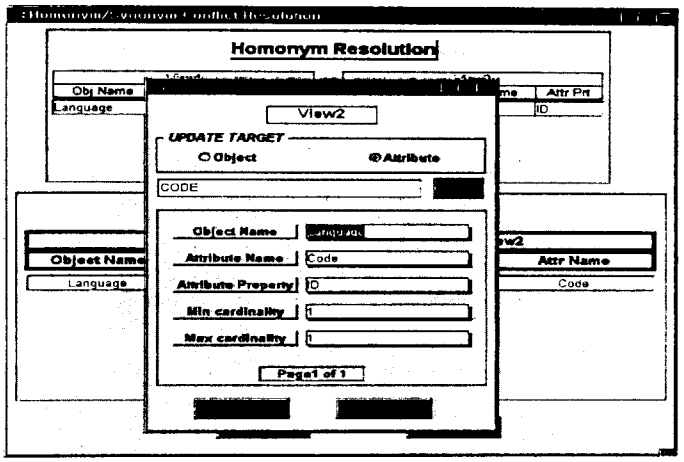
〈그림 4-2〉 뷰 친밀도 분석

이 화면에서 Continuous 버튼을 누르면 객체 친밀도 분석을 수행하며, 동일 객체간에 공통으로 가지고 있는 속성들의 수를 계산하여 그 결과를 〈그림 4-3〉에서 같이 보여 준다.



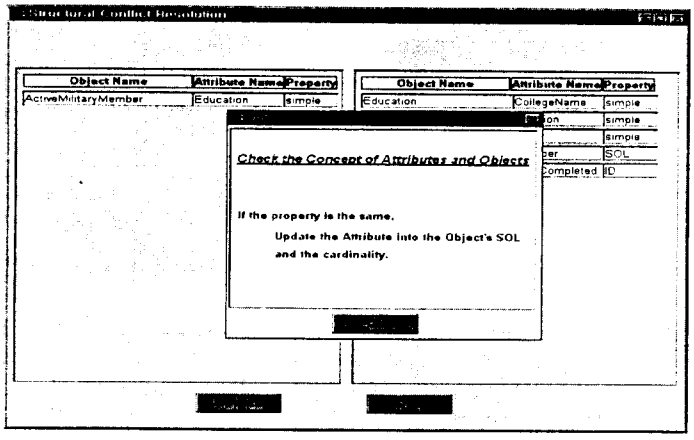
〈그림 4-3〉 객체 친밀도 분석

이 뷰와 객체 친밀도 분석을 기반으로 의미 충돌 해결과 통합을 수행하게 된다. 〈그림 4-3〉에서 OK 버튼을 누르면 의미 충돌 해결 시스템으로 들어간다. 이 시스템은 이름 충돌 해결과 구조적 충돌 해결을 다룬다. 이름 충돌 해결은 또다시 동음이의어와 이음동의어 해결로 구분된다. 이 과정은 뷰 간 객체가 동일한 항목을 보여주어 설계자가 객체의 속성에 해당하는 속성의 데이터 타입(Type), 크기(Size), 형태(Format) 등과 이에 해당하는 양식이나 레코드 형태의 인스턴스를 기반으로 판단할 수 있도록 한다. 또 이음동의어도 마찬가지로 유사한 객체를 화면에 보여 주어 동음이의어 해결과 같이 설계자가 최종 판단하도록 한다. 이 판단을 근거로 하여 UPDATE 버튼을 눌러 갱신을 한다. 화면은 〈그림 4-4〉에 나타나 있다.



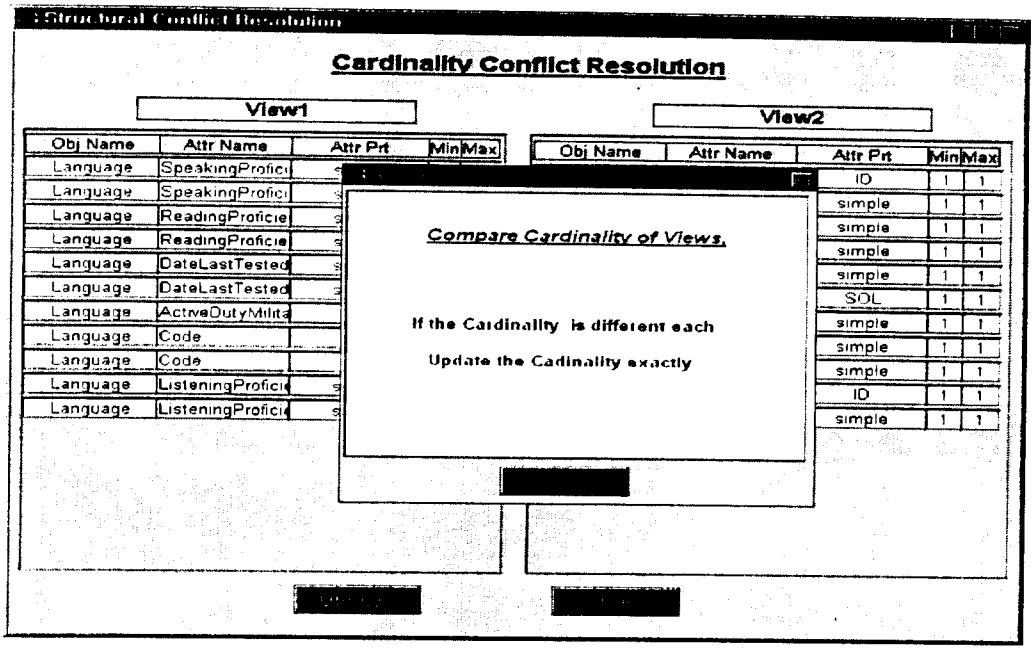
〈그림 4-4〉 동음이의어와 어음동의어 해결 화면

구조적 충돌에는 타입과 수제한 충돌이 있다. 타입 충돌은 두 뷰를 비교하여 동일개념이면 UPDATE 버튼을 클릭하여 뷰 데이터베이스를 갱신한다. 이는 〈그림 4-5〉에 표시하였다.



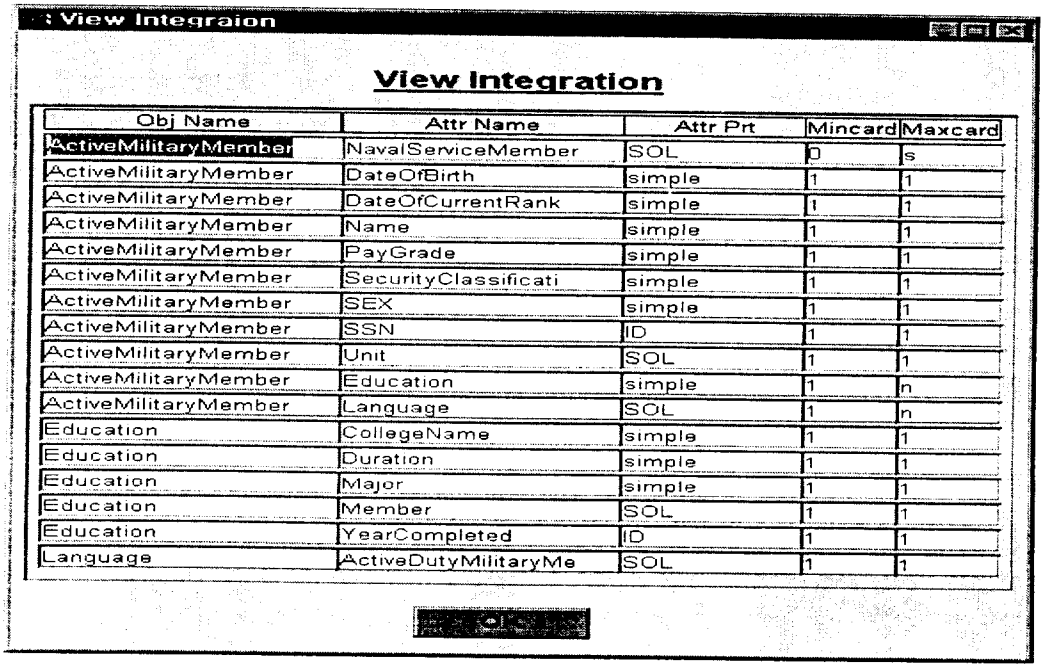
〈그림 4-5〉 타입 충돌 해결 화면

수제한 충돌 해결은 〈그림 4-6〉에서 보여 주고 있다.

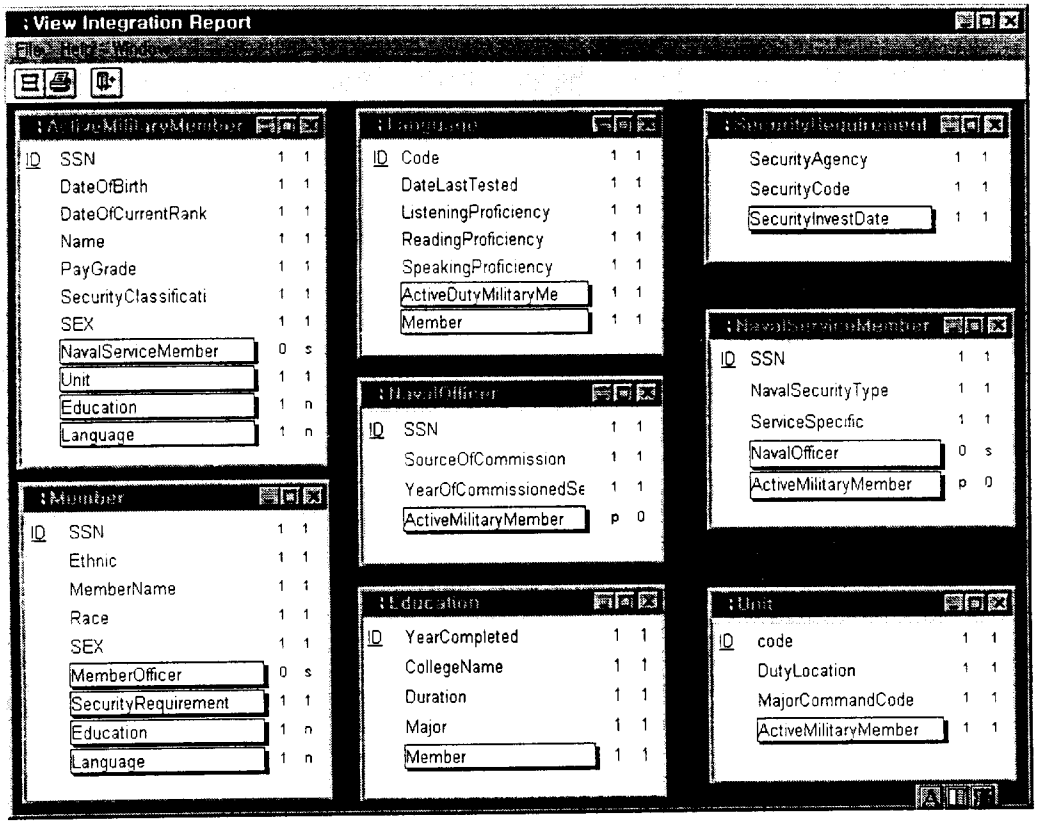


〈그림 4-6〉 수제한 충돌 해결 화면

의미 충돌 해결이 모두 완료되면 다음으로 통합 단계로 들어간다. 〈그림 4-7〉은 뷰 통합테이블이 보여지고, 여기서 OK 버튼을 누르면 〈그림 4-8〉과 같은 의미객체모델의 통합 뷰가 생성된다.



〈그림 4-7〉 뷰 통합 테이블



〈그림 4-8〉 통합 뷰

5. 결 론

본 논문에서는 뷰 통합을 위하여 의미객체모델을 이용하였고, 벡터공간을 이용한 뷰 간 친밀도를 계산하여 의미 충돌 해결과 통합을 용이하게 했다. 다양한 뷰를 생성한 요구 분석 정보(양식, 레코드 형태 등)를 최대한 활용하여 의미객체 식별자의 인스턴스를 파악하여 객체간의 관계를 규명하여 의미 충돌 해결을 할 수 있도록 했다. 또 파워빌더를 이용하여 분산 환경에서의 뷰들을 통합할 수 있도록 고려했다.

기존의 뷰 통합 방법론과 본 방법론과의 차이점의 첫째는 의미객체모델을 이용한 뷰 통합 방법론 개발이다. 둘째, 기존에는 뷰 합병 시 중요한 뷰를 설계자 주관에 의해 평가하던 것을 뷰간 친밀도 계산에 의한 평가 개념을 도입하여 객관적으로 평가할 수 있도록 했다는 점이다. 세째, 통합 절차에 뷰와 객체 간의 친밀도 분석을 통해 친밀도가 높은 뷰부터 의미 충돌 해결과 통합을 할 수 있도록 함으로써 합리적인 통합 방안을 제시하였다. 넷째, 의미객체의 이음동의어 및 동음이의어를 발견함에 있어 식별자와 식별자의 인스턴스를 최대한 활용하였다.

앞으로 본 논문은 다음과 같은 점에서 좀더 깊이 연구되어져야 한다. 첫째는 뷰 통합 지원 시스템의 완전 자동화 구현이다. 즉 설계자가 휴리스틱 방법에 의해서 이음동의어와 동음이의어임을 판별하는 과정의 자동화이다. 둘째는 통합 개념

뷰의 검증이다. 통합 개념 뷰의 최소성(Minimality), 이해성(Understandability) 등을 검증할 수 있는 평가 기준을 만드는 것이다.

참 고 문 헌

[1] Y. Kim and H. Lee, 의미객체모형을 이용한 개념적 데이터베이스 설계 지원 시스템의 개발, *KMIS* '95, 추계 학술 발표 대회 논문집, 1995, pp.705-726

[2] H. Kim and H. Lee, 양식을 이용한 분산 데이터베이스 설계 방법론, *KMIS* '95, 추계 학술 발표 대회 논문집, 1995, pp. 727-755

[3] K.T. Shin, N.G. Park, J.S. Park, and J.W.Park, 제조 데이터베이스 설계에서의 뷰 통합 방안, 경영과학 Vol.11, No.3, 11월, 1994

[4] C. Batini, S. Ceri, and S.B. Navathe, Conceptual Database Design: An Entity-Relationship Approach, Benjamin /Cummings, 1992, pp.119-133

[5] C. Batitni, and M. Lenzerini, A Methodology for Data Schema Integration in the Entity-Relationship Model, *IEEE Transactions on Software Engineering*, Vol. SE-10, No.6, Nov. 1984

[6] C. Batini, M. Lenzerini, and S.B. Navathe, A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Surveys*, Vol.18, No.4, Dec. 1986

[7] M.S. Bazaraa, and J.J. Jarvis, Linear Programming and Network Flows, John Wiley & Sons, 1977

[8] G. Booch, Object-Oriented Analysis and Design with Applications, Benjamin / Cummings, 1994

[9] S. Ceri, B. Pernici, and G. Wiederhold, Distributed Database Design Methodologies, *Proc. IEEE* Vol.75, No.5 1987, pp. 533-546.

[10] P.P. Chen, The Entity-Relationship Model: Toward a Unified View of Data, *Proceedings IFIPS Data Systems*, Vol. 1, No.1, Marh 1976, pp.9-37

[11] C.J. Date, An Introduction to Database System, Sixth Edition, 1995

[12] S. Dewitz, and M. Olson, Semantic Object Modeling With SALSA: A Case Book, McGrow-Hill, 1994

[13] R. Hull, and R. King, Semantic Database Modeling: Survey, Applications, and Research Issues, *ACM Computing Surveys*, Vol.19, No.3, Sept. 1987

[14] M. Kamel, Identifying, Classifying, and Resolving Semantic Conflicts in Distributed Heterogeneous Databases: A Case Study, *Journal of Database Management*, Vol.6, No.1, Winter 1995

[15] D.M. Kroenke, Database Processing: Fundamentals, Design, And Implementation, Fifth Edition, Prentice Hall 1995

[16] J. Martin, Information Engineering, Prentice Hall, 1989

[17] S.B. Navathe, R. Elmasri, and J. Larson, Integrating User Views in Database Design, *IEEE Computer*, Jan. 1986

[18] M.T. Ozsu, and P. Valduriez, Principles of Distributed Database System, Pren-

tice-Hall, 1991, pp.425-440

[19] M.P. Reddy, B.E. Prasad, P.G. Reddy, and A. Gupta, A Methodology for Integration of Heterogeneous Databases, *IEEE Transactions on Knowledge and Data Engineering*, Vol.6, No.6, Dec. 1994

[20] A.P. Sheth, S.K. Gala, and S.B. Navathe, On Automatic Reasoning for Schema Integration, *International Journal of Intelligent and Cooperative Information Systems*, Vol.2, No.1, 1993, pp. 23-50

[21] S. Spaccapietra, View Integration: A Step Forward in Solving Structural Conflicts, *IEEE Transactions on Knowledge and Data Engineering*, Vol.6. No. 2, April 1994

[22] V.C. Story, and R.C. Goldstein, Knowledge-Based Approaches to Database Design, *MIS Quarterly*, March 1993, pp. 25-46

[23] T.J. Teorey, Database Modeling And Design; The Entity-Relationship Approch, Morgan Kaufmann Publishers, 1990