속성 버전화에 기반한 시간지원 객체지향 모델의 형식화

이홍로* · 김삼남* · 류근호*

The Formalization of a Temporal Object Oriented Model Based on an Attribute versioning

〈요 약〉

객체지향 데이터베이스 시스템에서 시간지원 데이터베이스를 다룰 때 발생하는 중요한 문제는 관계 의미에 따라 시간과 속성을 결합하는 방법에 있다. 관계형 모델처럼 속성 버전화에 대한 기존의 연구 결과는 시간지원 객체지향 모델에 적용할 수 없다. 이것은 객체지향 모델이 복합 객체를 구성하기 위해서 기존의 모델보다 더욱 강력한 구성자들을 제공하기 때문이다. 그래서 이 논문은 객체지향 데이터베이스에 시간 개념을 통합하기 위한 형식적 접근방법을 제안한다.

이 논문의 목적은 객체 사이에 관계하는 일반화, 집단화와 연관화에 따라 시간지원 객체지향 데이터베이스 표현을 연구하는 것이다. 이 논문은 시간지원 객체지향 모델에서 속성 버전화의 개념을 정의하고, 객체 사이에 존재하는 관계에 대해서 시간을 표현하는 것에 중점을 둔다. 또한 관계 의미에 대한 제약조건을 규정하고, 표현 기준에 기반하여 검토한다. 이 논문은 객체지향 데이터 모델을 형식화함으로써 대수 연산자의 설계시 강력한 연산 기능을 제공할 뿐만 아니라 모듈의 재사용성을 제공할 수 있다.

* 충북대학교 자연과학대학 컴퓨터과학과

1. 서 론

시간지원 데이터베이스는 시간의 흐름에 따라 변화된 자료를 조작할 수 있는 데이터베이스이다.

기존의 데이터베이스 시스템은 실세계에서 발생한 사건에 대해서 오직 현시점에서 유효한 버전만을 관리하므로 과거의 이력자료가 필요할 경우에 로그 파일 등을 이용하여 별도의 처리 방법이 필요하며, 사용자에게 복잡한 질의 작성이라는 부담과 함께 처리 비용이라는 심각한 문제를 야기한다[Sno87, Sno94, Wuu92, Elm93b, 정경자96].

객체지향 모델은 실세계의 복잡한 사건의 계층적 구조를 규정하기 위해서 이 사건 사이의 관계에 대한 의미를 지원하는 강력한 구성자들을 제공한다. 시간이 추가된 객체지향 데이터베이스는 차세대 데이터베이스 시스템에 중요한 역할을 할 것이다. 그래서 이 데이터베이스의 특성을 완전하게 이용할 수 있고, 그 특성들이 통합될 수 있다면, 사용자에게 필요한 궁극의 데이터베이스를 제공하게 될 것이다. 시간지원 데이터베이스의 목적은 통합된 시스템에 과거, 현재와 미래 정보를 균일하게 결합하는 것이다[Sno87, Jen90, Elm90a, Wuu92, Pis92]. 객체지향 데이터베이스의 중요한 특성은 복합 객체에 적용할 수 있는 구조와 연산을 규정함으로써 복합 객체를 제공하는 것이다[Ber91, Deu91, Kim91].

시간지원 객체지향 데이터베이스는 과거 데이터에 대한 이력이 필요한 CAD/CAM, 통계적 데이터 처리, 법령의 저장, 감사 추적, 범죄 추적, 그리고 의료 분야 등에 이용될 뿐만 아니라 데이터 웨어하우징(Data Warehousing)에서의 이력 관리에도 효과적으로 이용된다. 시간에 종속적인 자료를 처리하기 위해서 시간지원 객체지향 데이터베이스에서는 클래스를 구성하는 객체의 각 속성과 메소드 중 시간 종속 데이터에 시간 의미를 부여하여 데이터를 관리하게 된다[Elm93a]. 또한 시간지원 데이터 모델과 언어에 대한 연구가 급속히 증가 추세에 있고, 관계형 데이터 모델에 기반해서 중점적으로 연구되어 왔고[Sno87, Gad88a, Nav89, Jen90], 객체지향 데이터베이스 모델에서는 좀더 좁은 범위에서 연구되어 왔다[Ela90b, Pis92, Wuu92].

객체지향 시스템 하에서 시간지원 데이터베이스를 다룰 때 발생하는 문제점은 시간을 객체와 연결할 것인가 아니면 속성과 연결할 것인가 이다. 이 시간지원 표현을 위한 결합 방법은 현실세계의 개체 및 관계에 대한 각 속성에 시간을 결합하는 속성 버전화와 시간속성을 추가하는 객체 버전화가 있다. 속성 버전화는 해당 타입 및 관계의 속성에 시간을 결합하여 이력 사항을 나타내는 것을 의미한다[Gad88a, Elm90b].

이 속성 버전화는 한 속성에 비시간 값과 시간 값을 동시에 같이 표현한 것이다. 객체 버전화는 해당 개체 타입 및 관계 타입의 비시간 속성에 시간 속성을 추가하여 이력 사항을 표현하는 기법이다[Nav89, Jen90, Sno94]. 이 버전화는 한 인스턴스에서 비시간 값에 시간 값을 결합한 것이다. 시간지원 개체-관계 모델은 개체-관계 모델의 의미를 바꾸거나 새로운 구성자를 추가함으로써 정보의 시간적인 면을 좀 더 자연스럽게 모델링 하는 것을 시도하고 있다. 관계형 데이터베이스 모델에서는 제일 정규형이나 비제일정규형에 의해서 영향을 받지만, 객체지향 데이터베이스는 복합 객체를 위한 강력한 구성자(constructor)를 지원하기 때문에 시간지원 객체지향 데이터베이스 정규화에 영향을 받지 않는다[Lec88].

그런데 지금까지 수행된 연구는 시간 개념이 포함된 객체지향 모델에서 관계 의미를 일반화, 집단화와 연관화로 구분하여 형식적으로 표현하는 기법에 대해서는 제시하지는 못하였다. 그래서 관계의 의미부여에 따라 의미를 집단화와 연관화로 구분하는 것이 필요한데, 일반적인 객체지향 모델은 개체 사이의 관계성을 모델화하기 위해서 두 가지 구성자만을 제공한다. 이것은 객체의 합성을 모델화하기 위한 집단화 참조와 객체 사이에 임의의 연관화를 모델화 하기 위해

사용되는 연관화 참조는 의미와 구문을 나누지 않았기 때문에 객체의 합성과 객체의 임의의 연관화 사이에 애매모호성을 증가시킨다. 또한 합성 객체와 성분 객체 사이에 구조와 행위의 상호 작용에 대해서 오류를 발생시킬 수 있다.

또 다른 문제점은 개체 관계의 일반화와 집단화에 대한 상속성의 지원 개념이 부족하다. 즉 부분 클래스 상속과 이에 대한 구현은 많은 연구를 하였으나[Ame90, Neu89], 집단화 계층의 상속은 거의 연구되지 않았다[Lin93, Lee94, Lee95, 김삼남97a, 김삼남b]. 시간 차원 데이터베이스 문제점은 시간 표현 방법, 사실과 시간의 연결방법, 그리고 속성 값 표현 방법에 기반으로 하여 연구해야 한다.

이 연구는 유도형 시간 속성을 추상화 클래스화시켜 속성-영역에서 시간 영역에 도입하여 일반화, 세부화, 집단화와 연관화의 의미에 따라 메소드 및 속성에 시간을 도입하여 클래스 다이어그램과 클래스 정의어 구문을 규정하고자 한다.

집단화 참조와 연관화 참조 사이의 일반적 차이에 기반을 두고 있으며 세분화와 집단화 추상성 둘 다에 상속성을 도입함으로써 기능면에서 재사용성을 확장하고 있다. 또한 시간지원 객체지향 모델에 객체지향 개념을 도입하여 속성 버전화를 정의하고 객체 사이에 시간 관계성에 대해 다양한 표현을 하고자 하며, 속성 버전화에 유효시간과 거래시간을 도입한다. 결국, 이 논문은 객체지향 데이터 모델을 형식화함으로서 질의 대수 연산자를 설계시 강력한 연산 기능을 제공할 뿐만 아니라 모듈의 재사용성을 증가시킬 수도 있다.

이 논문의 내용을 효율적으로 전개하기 위하여 제 2장에서는 시간지원 모델과 객체지향 모델을 규정한다. 그리고 제 3장에서는 속성 버전화를 위한 시간지원 객체지향 모델을 제안하며, 제 4장에서는 제시한 모델에 대한 시간 제약 조건을 규정하고 표현 기준에 대해 검토한다. 그리고 마지막으로 결론을 내리고 향후 연구를 토의한다.

2. 시간지원 모델과 객체지향 모델

제 2.1절에서는 시간 개념을 모델화하기 위해서 시간 표현 방법, 시간 결합 방법과 시간차원에 대해 설명한다. 제 2.2절에서는 객체지향 모델을 규정하고, 이에 대한 예를 제 2.3절에서 보인다.

2.1 시간의 정형화

시간을 정형화하기 위해서는 시간 정의역을 표현하는 방법, 시간 값을 해당 개체 및 관계에 결합하는 방법, 그리고 이력 정보를 처리하기 위한 모델이 지원되어야 한다.

가. 시간 표현

시간 표현 방법은 연속적인 시간을 표현하기 위해서 이산 시점, 시간 간격, 그리고 시간을 집합화하기 위한 시간 요소가 있어야 한다[Jen90, Sno94]. 이들을 구체적으로 정의하면 다음과 같다.

【정의 2.1】 시점

시점은 시간을 하나의 점으로 표현한 것을 의미한다.

$$T = \{t_{-\infty}, \ldots, t_0, t_1, \ldots, t_{now}, t_{now+1}, \ldots, t_{\infty}\}$$

여기서, T는 전 순서 이산 시점(total order discrete time point)들의 셀 수 있는 무한 집합이며, t는 연속적으로 증가하는 시점을 표현한 것으로 t_{now}는 현재 시점을 나타내는 시간의 단위(chronons: 초, 분, 시, 일, 주, 월, 분기, 반기, 년 등)이다. 또한 Tp = <T,< >, Tp는 시점 정의역이고, <는 전순서를 나타낸다. □

【정의 2.2】 시간 간격

시간 간격(time interval: TI)은 두 사건(event) 사이의 기간을 의미한다. 이 간격은 정의 2.1로 규정된 연속적인 두 시점 t_{is}와 t_{ie} 순서쌍의 집합이다.

$$TI = \{ti_0, ti_1, ..., ti_n\}$$

여기서는 ti_i는 $(t_{is}, t_{ie}]$, $[t_{is}, t_{ie})$, $[t_{is}, t_{ie}]$ 또는 (t_{is}, t_{ie})이며, $1 \le i \le n$ 이다. 세부적으로 "("와 ")"는 인접 시점을 포함하고, "["와 "]"는 인접 시점을 포함하지 않는다. 또한 t_{is}는 시작 시점을 나타내고, t_{ic}는 종착 시점을 나타내며, 그리고 이들은 시간의 전체 집합 T에 존재하는 시점들이다.□

【정의 2.3】 시간 요소

시간 요소(time element: TE)는 이력 시간을 시점이나 시간 간격 단위로 집합화하여 표현하기 위한 기법이다. 그래서 시점들에 의한 요소들의 집합을 사건 시간 스탬프 및 사건 생명주기라 한다. 또한 시간 간격들에 의한 요소들의 집합을 간격 시간 스탬프 및 간격 시간 사건 생명주기라 한다.

$$TE = \{te_1, ..., te_i, ..., te_n\} \subseteq T/TI$$

여기서 te_i는 시점 t_i 및 시간간격 ti_i를 나타내며, $1 \le i \le n$ 이다. □

그리고 시간지원 데이터베이스 시스템을 위한 시간 차원은 유효시간, 거래시간, 이원시간, 그리고 사용자 정의시간으로 구분되며, 다음과 같이 규정할 수 있다.

【정의 2.4】 유효시간

유효 시간(valid time: vt)은 사실에 대한 논리적인 시간을 의미하며, 어떤 사실이 발생하거나 소멸된 시간을 나타낸다[Tan86, Tuz90]. 일반적으로 사용자에 의해서 유효시간이 결정되며, 사용자가 위치한 장소의 시간 시스템에 의해 상대적 시간 값으로 표현된다.

$$vt \subseteq T/TI$$

여기서 T는 이산 시점의 전체 집합이며, TI는 시간 간격의 전체 집합이다. □

【정의 2.5】 거래 시간

거래 시간(transaction time: tt)은 어떤 사실이 데이터베이스관리시스템에 의해서 처리되어진 시간을 의미하며, 시스템 클럭(system clock)에 의해 자동적으로 삽입된다[Sno87].

$$tt \subseteq T/TI$$

여기서 T는 이산 시점의 전체 집합이며, TI는 시간 간격의 전체 집합이다. □

【정의 2.6】 이원 시간

이원 시간(bitemporal: bt)은 유효시간과 거래 시간을 함께 표현한 것이다[Gad88b].

$$bt = \langle vt, tt \rangle \subseteq T/TI$$

여기서 vt는 유효시간이고, tt는 거래시간이다. □

【정의 2.7】 사용자 정의 시간

사용자 정의 시간(user defined time: ut)은 유효 시간과 거래 시간에 의해 처리되지 못하는 시간으로 사용자가 이 시간을 직접 정의하여, 부가적으로 내부에 표현하는 방법과 입출력 함수가 요구된다.[Sno87].

$$ut \subseteq T \quad \square$$

나. 시간 결합

시간 결합 방법은 현실 세계의 개체 및 관계에 대한 이력 자료를 각 속성에 시간을 결합하는 속성 버전화와 시간속성을 추가하는 객체 버전화가 있다. 이러한 시간 결합 방법을 수식으로 표현하면 다음과 같이 정의할 수 있다.

【정의 2.8】 속성 버전화

속성 버전화(attribute versioning: av)는 해당 클래스의 속성에 시간을 결합하여 이력 사항을 나타내는 것으로, 비시간 값과 시간 값을 한 속

성에 표현한다[Gad88a, Tan86].

$$av_i = \langle v_i, bt \rangle$$

여기서 av_i는 한 타입의 전체 속성 $AV = \langle av_1, ..., av_i, ..., av_n \rangle$ 중에서 한 버전 속성을 나타내고, 이 v_i는 속성 av_i의 비시간 값이며, bt는 이원시간 값이다. □

【정의 2.9】 객체 버전화

객체 버전화(object versioning: tv)는 해당 객체의 비시간 속성에 시간 속성을 추가하여 이력 사항을 표현하는 기법으로, 한 인스턴스에 비시간 값과 시간 값을 함께 결합한 것이다[Gad88b, Sno87].

$$tv = \{v_1, ..., v_i, ..., v_n, bt, ut\}$$

여기서 v_i는 한 인스턴스의 비시간 값들이다($1 \le i \le n$). bt는 이원 시간 값이고, ut는 사용자 정의 값이다. □

2.2 객체-지향 모델

기본적으로 실세계의 각 사건은 식별이 가능한 객체로 모델화 된다. 각 객체는 속성 값의 집합과 메소드의 집합으로 구성된다. 같은 구조와 행위를 공유하는 객체들은 클래스로 모아진다.

이 클래스의 객체들은 같은 객체 타입 정의를 가진다. 덧붙여서 객체지향 모델은 객체 식별성, 메소드, 추상 자료형, 캡슐화, 클래스, 클래스 계층구조 및 일반화 상속 등과 같은 공통적으로 수용하는 객체지향 개념을 지원한다[Lec88, Kim91]. 또한 관계 의미의 역할에 대응하는 집단화와 연관화를 클래스 계층 구조로 나누어지며, 다형성 관점에서 재사용성을 일반화 상속성과 집단화 상속으로 구분하여 기술한다[Ling93, Lee94, 김삼남97]. 객체 식별자는 객체의 존재를 의미하며, 값은 주어진 클래스의 객체를 표현하는 것이다. 객체들은 응용할 수 있는 메소드의 집합인 인터페이스를 통해서만 접속할 수 있다. 더 자세한 객체지향 데이터베이스의 특성과 장점에 대해서는 문헌[Ber91, Deu91, Elm93a]을 참조하며, 여기서는 필요한 구성요소만을 기술한다.

클래스의 속성 및 메소드에 대한 타입 생성자들은 기본 타입, 튜플 타입, 그리고 집합 타입으로 구분된다. 기본 타입은 정수, 실수, 문자열, 부울, 등이다. 튜플타입은 $\langle A_1 : \tau_1, \cdots, A_n : \tau_n \rangle$이며, 여기서 $\tau_1, \cdots, \tau_n$은 타입들이고, $A_1, A_2, \cdots, A_n$이 속성이름들이다. 집합 타입은 $\{\tau\}$이다. 또한 τ 가 타입이면, ref τ 는 참조타입이다. 그래서 속성과 메소드로 구성되는 클래스 스킴은 다음과 같이 규정할 수 있다.

가. 클래스 스킴

타입에 대해서 유사한 객체들을 모아서 저장하고 생성하기 위한 기틀로써 속성과 메소드로 구성되는 클래스 스킴은 $C = (A, M)$이다. 여기서 클래스의 속성을 나타내는 $A = \{A_c \cup A_s \cup A_d\}$는 표현 방법에 따른 속성들의 전체 집합이다.

세부적으로 속성 A_c는 객체의 값에 대한 데이터 타입의 복잡성에 의해서 단순 속성(simple attribute)과 합성속성(composite attribute)으로 구분되고 A_s는 객체들의 다중성에 의해서 단일 값 속성(single valued attribute)고 다중 값 속성(multi valued attribute)으로 구분된다. 그리고 A_d는 데이터베이스에 값 및 객체들의 저장성 여부에 의해서 저장 속성(stored attribute)과 유도속성(derived attribute)으로 구분된다. 또한 데이터베이스에 저장된 객체 및 $M = \langle mn, ms, mb \rangle$ 값을 변경하거나 생성하는 기능을 하는 메소드는로 구성되며, mn은 메소드 이름,ms $f : S \times P_1 \times \cdots\cdots \times P_k \rightarrow RT$로 표기되는 메소드 요약이며, 여기서 S는 메소드 m이 정의되는 초기 입력 인자의 클래스이고,

$P_i(i=1, \cdots, k)$는 메소드 인자들이다. 또한 RT는 출력 자료 클래스이고, mb는 메소드 m의 구현과 의미를 규정하는 몸체이다. 개념적으로 메소드는 한 클래스와 연관된 함수이다. 클래스 계층 구조를 일반화, 집단화 그리고 연관화에 따라 관계의미를 규정하면 다음과 같다.

나. 일반화 클래스 계층, 집단화 클래스 계층과 연관화 클래스 계층

관계의미가 일반화 클래스 계층(generalization class hierarchy)인 경우에 τ 와 σ 가 전체 클래스 C에 대한 한 클래스라면, $\tau \leq \sigma$ 로 표기되는 σ 와 τ 의 부분 클래스화는 다음처럼 재귀적으로 정의된다.

【정의 2.10】 부분 클래스화 및 일반화 상속

(i) τ_b가 기본클래스라면 $\tau_b \leq \tau_b$

(ii) 집합이나 참조 구성자에 대한 클래스에서 $\tau \leq \sigma$ 이면 $\{\tau\} \leq \{\sigma\}$ 과 ref $\tau \leq$ ref σ

(iii) 튜플 구성자에 대한 클래스 τ 와 σ 에 대해서 $\tau_i \leq \sigma_i, i=1,n$이면 $(A_1: \tau_1, \cdots, A_n: \tau_n) \leq (A_1: \sigma_1, \cdots, A_m: \sigma_m)(n \leq m)$ □.

위 정의 2.10에 의해서 부분클래스는 부분 순서(partial order)를 이룬다[Car85]. 일반화 상속은 클래스화에 기반한 클래스 합성기법이다. 그래서 클래스 τ 에 의해서 규정된 모든 속성들이 클래스 σ 에서도 사용할 수 있다면 $(\tau \leq \sigma)$, 클래스 τ 로부터 클래스 σ 로 속성 및 메소드를 상속한다고 한다. 하나 이상의 상위 클래스부터 한 부분클래스가 규정된다면, 다중 상속이라 한다.

또한 부분 클래스의 개수가 두 개 이상일 경우에 이 각 객체들의 중복성 여부에 따라 중복 상속(IS-A(o))과 분리 상속(IS-A(d))으로 나눌 수 있다[Elm93b]. 두 클래스 (σ, X_σ)와 (τ, X_τ)사이에 부분 클래스 관계가 존재한다면, $\sigma, \tau \in C$에 대해서 $\tau \leq \sigma$(내포성)와 $X_\tau \geq X_\sigma$(외연성)가 성립한다.

관계의미가 집단화일 경우에는 두 객체 사이의 관계가 합성 객체와 성분 객체일 때를 의미하며, 집단화 클래스 계층(aggregation class hierarchy)을 이룬다. 연관화일 경우에는 두 객체가 임의로 연결되었을 때를 의미하며, 연관화 클래스 계층(association class hierarchy)을 이룬다[Lin93, Lee95].

【정의 2.11】 집단화 클래스 계층

C가 클래스의 집합이며, O가 클래스 C에 연관된 객체의 집합이라 하자. O에 대한 집단화 클래스 계층(C, ≒)은 C로 표현하고 집단화 참조관계는 ≒로 표현되는 클래스구조이다. □

정의 2.11에 의해서 집단화는 성분 클래스들을 튜플 구성자들에 의해 합성 클래스로 추상화하는 방법이다. 이들 개체 사이에서 속성과 영역 관계를 형성할 때 합성 참조(composite reference)라 한다.

합성 참조는 성분 클래스가 합성 클래스의 존재에 따라 생존 유무를 독립/종속(Independent/Dependent) 으로 분류한다. 그리고 성분 클래스가 단지 하나의 합성 클래스와 관련이 있는지와 다수의 합성 클래스와 관계가 있는지에 따라 독점/공유(Exclusive/Sharable)로 구분되며, 합성 클래스로부터 속성과 메소드의 상속 여부도 관계된다.

이와 같은 차원의 분류를 조합하면 4가지의 합성 참조 형태를 얻을 수 있다.[Kim91].

【정의 2.12】 연과화 클래스 계층

C가 클래스의 집합이며, O가 클래스 C에 연관된 객체의 집합이라 하자. O에 대한 연관화 클래스 계층(C, ≒)은 C로 연관화 참조관계 ≒로 표현되는 클래스구조이다. □

그래서 정의 2.12에 의해서 연관화 관계는 임의의 객체들이 임의로 연결 할 수 있으므로 양방향성 속성-정의역 관계를 형성한다. 또한 정의 2.11에 의해서 집단화 클래스는 종속성, 추이성과 단방향성을 가지므로 다음과 같이 집단화 상속을 정의할 수 있다.

【정의 2.13】 집단화 상속

클래스 τ 와 σ 사이에 집단화 참조 ($\sigma \sqsubset \tau$) 관계는 부분 순서관계를 형성하고 시스템 정의 클래스인 OBJECT에 의해서 반격자 구조를 이룬다. 이 집단화 참조는 단방향으로 강한 결집력을 가진다. 그래서 집단화 참조에 기반한 클래스 합성 기법은 성분 클래스 σ 의 속성들은 클래스 τ 에 구문상(syntactic)으로 재사용할 수 있으며, 강제성(coercion)에 의한 다형성을 이룬다 [Ling93, Lee93]. 이런 상속성을 집단화 상속성이라 한다. 두 클래스 (σ, X_σ)와 (τ, X_τ)사이에 집단화 관계가 존재한다면, $\sigma, \tau \in T$에 대해서 $\tau \leq \sigma$ (내포성)와 $X_\sigma \leq X_\tau$ (외연성) 가 성립한다. □

다음의 2.3절에서는 지금까지 기술한 정의들을 바탕으로 객체지향 데이터베이스의 스키마를 구성한다.

2.3 객체지향 데이터베이스 스키마의 예

클래스 계층 구조는 속성-영역 관계에서 참조되는 영역의 "null" 값 유무에 따라 참여 제약조건을 가지며, 참조되는 영역의 다중성 여부에 따라 집합 구성자를 가질 수 있다. 일반화 상속은 부분 클래스의 중첩성 여부에 따라 명시적으로 IS-A(o)와 IS-A(d)를 표현할 수 있다. 또한 집단화 상속에 의해서 명시적으로 CONSTITUENT-OF를 표현할 수 있다. 그래서 이 객체지향 모델의 개념을 바탕으로 프로젝트 도서관리 데이터베이스 스키마의 예를 설계하면 다음 그림과 같다.

```
class PERSON = (
        pname                   : string
        birthday                : date;
        home_address            : CITY_ADDRS, independent & sharable, total
end class PERSON
class EMPLOYEE = (
        IS_A                    : PERSON, total;
        hiredate                : date;
        salary                  : real;
        works_for               : {<dept of : ref DEPARTMENT, Hours: real>};
        works_on                : {ref PROJECT};
        affiliates              : ref COMPANY, total;
        method void give-raise(percent : real););
end class EMPLOYEE;
class COMPANY = (
        cpname                  : string;
        location                : CITY_ADDRS, independent & sharable, total
        president               : {ref EMPLOYEE}, total;
        staff                   : {ref EMPLOYEE}, total
        projects                  {ref PROJECT};);
end class COMPANY;
class CITY_ADDRS = (
        constituent_of          : COMPANY, PERSON;
        cityname                : string;
        zipcode                 : string;);
end class CITY_ADDRS;
class DOCUMENT = (
```

```
                docno           : integer;
                title           : string;
                keyword         : string;
                authors         : {ref EMPLOYEE};
                contribute_to   : ref PROJECT ; //inverse of Dept.documents
end class DOCUMENT;

class PROJECT = (

                contract#       : integer;
                proj_name       : string;
                leader          : ref EMPLOYEE
                budget          : integer;
                documents       : {ref DOCUMENT};);
end class PROJECT;

class INTERNAL_PROJECT = (
                IS_A(o)         : PROJECT, total
                IPno            : integer;)
end class INTERNAL_PROJECT;
class FUNDEDATION PROJECT = (
                IS_A(o)         : PROJECT, total;
                FPno            : integer;)
end class  FUNDED_PROJECT;
class FUNDEDATION PROJECT = (
                IS_A(d)         : FUNDED_PROJECT, total
                FOPno           : integer;)
end class  FUNDED_PROJECT;
class CORPORATE_PROJECT = (
                IS_A(d)         : FUNDED_PROJECT, total
                COPno           : integer;)
end class  CORPORATE_PROJECT;
```

<그림 2.1> 객체지향 데이터베이스 스키마

위 <그림 2.1>에서 개체 클래스들은 CITY_ADDRS, COMPANY, PERSON, EMPLOYEE, PROJECT, DEPARTMENT 등이다. 개체 클래스 PERSON에서 birthday는 원자 속성이고, home_address: CITY_ADDRS, independent & sharable, total에서 home_address는 집단화 속성을 나타내고, CITY_ADDRS는 집단화 참조되는 클래스이며, independent & sharable은 합성 클래스에서 독립적이고 공유할 수 있는 상속성을 의미하며, total은 속성-영역 관계에서 참조되는 영역이 필수적으로 존재함을 의미한다. 개체-관계 모델에서 전체 참여조건을 의미한다. staff: {ref EMPLOYEE}, total에서 staff와 연관화 속성을 나타내며 "{}"는 참조되는 영역의 다중성 여부에 따른 집합 구성자를 나타낸다. staff와 president는 COMPANY와 EMPLOYEE 사이의 역할에 의한 연관화 속성들이다. INTERNAL_PROJECT와 FOUNDATION_PROJECT 클래스에서 IS-A(o)와 IS-A(d)는 각각 일반화 상속성에서 중복성과 분리성에 의한 제약조건을 나타낸다.

3. 시간지원 객체지향 모델

이 장에서는 시간 개념이 추가된 객체지향 클래스를 기술하고, 속성 버전화를 위한 클래스 정의어 부문과 다이아그램을 규정한다. 그리고 이

에 대한 예를 설명한다.

3.1 시간지원 클래스와 객체

기본 자료형에 대한 정의역의 전체 집합 U_D, 전체 정의역 D, 객체 식별자의 전체 집합 ID이라 하자. 비시간 속성의 전체 집합은 $U_A = \{A_1, A_2, \cdots, A_{nd}\}$이라 하자 그리고 시간 속성의 전체 집합은 $U_A = \{vta, tta, uta\}$이라 하며, 여기서 vta는 유효시간의 속성이라 하고, tta는 거래 시간의 속성이라 하며, uta는 사용자 정의 시간이라 하기로 하자.

시간지원 객체지향 속성 버전화를 위한 개체 클래스, 관계 클래스, 속성, 역할, 제약조건 및 정의역을 위한 스킴은 다음과 같다.

【정의 3.1】 객체지향 속성 버전화를 위한 클래스 스킴 $C = \langle C_E, C_R, \sigma, \rho, D, F, AT \rangle$이다.

개체 클래스의 유한 집합 C_E

$$C_E = \{C_{E1}, \ldots, C_{Ei}, \ldots, C_{En}\}$$
$$= C_{RE} \cup C_W = \{E_{RE1}, \ldots, E_{REp}, \ldots, E_{REs}\} \cup \{E_{W1}, \ldots, E_{Wq}, \ldots, E_{Wt}\}$$

여기서 C_{RE}는 명시적 키를 가지는 개체 클래스이고, C_W는 명시적 키를 가지지 않는 개체 클래스이다.

관계 클래스 유한 집합 C_R

$$C_R = \{C_{R1}, \ldots, C_{Ri}, \ldots C_{Rn}\}$$

여기서 C_R은 관계 클래스이다.

관계 클래스 이름의 요약(signature) σ

$$\sigma: C_R \rightarrow C_E{}^{\times}$$

여기서 관계 클래스 C_R에 의해서 관계되는 개체 클래스는 $C_E{}^{\times} = C_{E1} \times \ldots C_{Ei} \times \ldots \times C_{En}$이다.

· 역할 할당 함수 (ρ)

$$\rho: R \rightarrow (SE_i \rightarrow role_i)$$

여기서 SE_i는 $E^{\times}$에 대한 관계 R의 n-항 관계에서 관계의 의미를 나타내며, 이 관계 의미는 일반화(S_{gc}) 및 세분화(S_{sp}), 집단화(S_{ag}), 그리고 연관화(S_{as})로 구분된다. 각 SEi에 대해서 관계 R의 역할은 $role_i = \{role_{ge}/role_{sp}, role_{ag}, role_{as}\}$로 구분되며, 이 역할은 한 개체가 관계 R에 의해서 다른 n-1의 개체와 관계하는 기능을 나타낸다.

기본 객체들의 유한 집합 D

$$D = \{D_1, \ldots, D_i, \ldots, D_n\}$$

여기서 기본 데이터 타입의 값에 대응하는 정의역이다.

속성 타입 함수 AT

$$AT: F \rightarrow \{(C_{RE} \cup C_W) \rightarrow D^{\times} \times T \cup C^{\times} \times T \cup \langle D, ATC \rangle^{\times} \cup \langle C, ATC \rangle^{\times}\}$$

여기서 전체 속성 이름 $F = \{F_A, F_{TA}, uta\}$에서 F_A는 전체 비시간 속성이름이고, 이원시간 속성이름 $F_{TA} = \{vta, tta\}$는 유효시간 속성이름(vta)과 거래시간 속성이름(tta)을 조합시킨 것이며, uta는 사용자 정의 시간 속성이름이다. F_A에 대한 정의역 ND는 원자 객체의 정의역 D와 비시간 합성 클래스 C의 조합이다. 즉 $ND = D^{\times} \cup C^{\times}$이다. 여기서 $D^{\times} = D_1 \times \ldots \times D_i \times \cdots \times D_m$이며, $C^{\times} = C_1 \times \ldots \times C_j \times \ldots \times C_n$이다. F_{TA}에 대한 정의역들은 원자 객체에 대한 시간 속성에 대한 정의역들 $\langle D, ATC \rangle^{\times} = \langle D_1, ATC \rangle \times \ldots \times \langle D_i, ATC \rangle \times \ldots \times \langle D_m, ATC \rangle$과 합성 객체에 대한 정의역들 $\langle C, ATC \rangle^{\times} = \langle C_1, ATC \rangle \times \ldots \times \langle C_j, ATC \rangle \times \cdots \times \langle C_n, ATC \rangle$ 으로

나누어진다. 여기서 ATC는 이원시간에 대한 정의역을 추상 클래 $D^{\times} \times T$스화한 것이다. 또한 uta의 값의 집합은에서 T이다. □

속성 버전화를 위한 시간지원 객체지향 모델의 클래스를 속성-영역에 의해 스킴을 정의하면 다음과 같다.

【정의3.2】 개체 클래스 스킴

$C_{OGE} = \langle A, OID, \triangleq, \asymp, \cong \rangle$

- 속성들의 집합: $F = \{F_A, F_{TA}, uta\}$이다. 여기서 $F_A \subseteq U_A, F_{TA} \subseteq U_{TA}, uta \subseteq U_{TA}$ 이다.
- 식별자들의 집합: OID ⊆ ID이다.
- 원자 객체 정의역의 집합: DOM: $F_A \cup uta \dashrightarrow U_D \cup \{T\}$
- 일반화 클래스 계층 $\triangleq \subseteq C \times C$
- 집단화 클래스 계층 $\asymp \subseteq C \times C$
- 연관화 클래스 계층 $\cong \subseteq C \times C$ □

【정의 3.3】 관계 클래스 스킴

$C_{OGR} = \langle F, OID \rangle$

- 속성들의 집합: $F = \{F_A, F_{TA}, uta\}$이다. $F_A \subseteq U_A$, $F_{TA} \subseteq U_{TA}$, $uta \subseteq U_{TA}$이다.
- 식별자들의 집합: OID ⊆ ID이다.
- 원자 객체 정의역의 집합: DOM: $F_A \cup uta \dashrightarrow U_D \cup \{T\}$ □

속성 버전화를 위한 클래스 데이터베이스의 스킴은 클래스 스킴의 유한 집합으로 실체 클래들과 클래스의 합이며 이에 대한 정의는 다음과 같다.

【정의 3.4】 실체 객체(O_{OGE})

O_{OGE}: (A(S_i) $\dashrightarrow$ D) ∪ (O_{OGE} (S_i): $O_{OGE)}$.1 $\dashrightarrow$ D) ∪ (O_{OGE} T(S_i): $O_{OGE)}$.1 $\dashrightarrow$ Component Class) ∪ (O_{OGE} T(S_i): $O_{OGE)}$.1 $\dashrightarrow$ Associated Class)

여기서 1은 개체 객체에 대한 생명주기이며, 이에 대한 정의역은 정의 2.3에 의해서 TE이다. □

속성 버전화를 위한 관계 객체에 대한 정의는 다음과 같다.

【정의 3.5】 관계 객체(O_{OGE})

O_{OGE}: (A(S_i) $\dashrightarrow$ D) ∪ (O_{OGR}(S_i): $O_{OGR)}$.1 $\dashrightarrow$ D) ∪ (O_{OGR} T(S_i): $O_{OGR)}$.1 $\dashrightarrow$ Entity Domain Class)

여기서 1은 관계 객체에 대한 생명주기이며, 이에 대한 정의역은 정의 2.3에 의해서 TE이다. □

3.2 속성 버전화를 위한 클래스 정의어 구문

정의 3.1에 의한 속성 버전화는 객체 클래스에 의한 구문과 관계 클래스에 의한 구문으로 나누어진다. 그래서 정의 3.2에 의해서 속성 버전화를 위한 개체 클래스의 구문을 규정하면 다음과 같다.

CLASS class_name =
{

IS-A : superclass names **IS-A(d)** : disjoint constraint superclass names **IS-A(o)** : overlapping constraint superclass names (generalization inheritance)
CONSTITUENT-OF : composition class names (aggregation inheritance)

attribute $name_1$: [{}basic domain class[}]
.
.
.
attribute $name_m$: [{}basic domain class[}]
user defined time : [{}**T**{}]
(non_temporal atomic attributes & user defined time attribute)

tattribute $name_1$: list<tuple($attribut_1$: [{}basic domain class[}], bt : ATC)>
.
.
.
tattribute $name_n$: list<tuple($attribut_n$: [{}basic domain class[}], bt : ATC)>
(temporal atomic attributes)

tagattribute $name_1$: list<tuple($agattribut_1$: [{}component domain $class_1$[}], bt : ATC)>
.
.
.
tagattribute $name_0$: list<tuple($agattribut_0$: [{}component domain $class_0$[}], bt : ATC)>
(temporal aggregation relationship attributes)

tasattribute $name_1$: list<tuple($asattribut_1$: [{}**ref** associated domain $class_1$[}], bt : ATC)>
.
.
.
tasattribute $name_q$: list<tuple($asattribut_q$: [{}**ref** associated domain $class_q$[}], bt : ATC)>
(temporal association relationship attributes)

}

<그림 3.1> 속성 버전화에 대한 개체 클래스 구문

위에서 기술한 정의 3.2에 의해서 속성 버전화를 위한 관계 클래스의 구문을 규정하면 다음과 같다.

CLASS class_name = {

ratribute $name_1$: [{}basic domain class[}]
.
.
.
rattribute $mame_m$: [{}basic domain class[}]
user defined time : [{}**T**{}]
(non_temporal atomic relationship attributes & user defined time attribute)

trattribute $name_1$: list<tuple($ratribut_1$: [{}basic domain $class_1$[}], bt : ATC)>
.
.
.
trattribute $name_n$: list<tuple($ratribut_n$: [{}basic domain $class_0$[}], bt : ATC)>
(temporal atomic relationship attributes)

eattribute $name_1$: list<tuple($eattribut_1$: [{}**ref** entity domain $class_1$[}], bt : ATC)>
.
.
.
eattribute $name_o$: list<tuple($eattribut_q$: [{}**ref** entity domain $class_q$[}], bt : ATC)>
(referenced entity attributes)

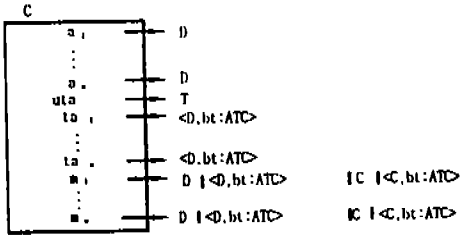
}

<그림 3.2> 속성 버전화에 대한 관계 클래스 구문

<그림 3.1>과 <그림 3.2>에 대한 자세한 내용은 다음의 3.3절에서 다이어그램으로 나타내고, 제 3.4절에서 속성 버전화의 객체지향 스키마의 예를 설명한다.

3.3 속성 버전화의 다이어그램

제 2.1절에서 기술한 시간 모델과 제 2.2절의 클래스 계층구조에 기반하여 제 3.2절의 객체 사이의 관계 의미에 따른 속성-영역에 의해서 속성 버전화를 위한 시간지원 객체지향 다이어그램을 유도할 수 있다[Che76, Elm93a, 김삼남97a, 김삼남b]. 우선 일반화, 집단화와 연관화 참조 관계를 가지지 않는 클래스의 속성-영역 관계를 <그림 3.3>과 같이 표현할 수 있다.



<그림 3.3> 속성 버전화에서 관계 의미를 가지지 않는 클래스의 속성-영역

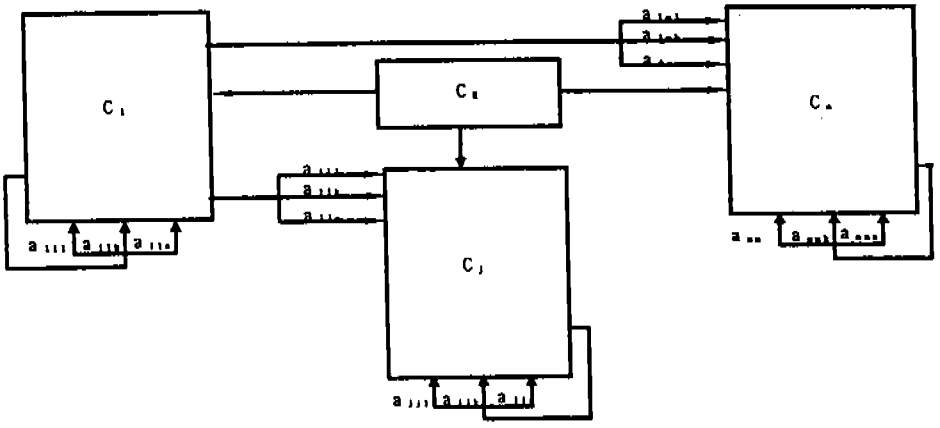
<그림 3.3>은 일반화, 집단화와 연관화를 가지지 않는 클래스의 속성과 메소드를 기술할 것이다.

여기서 a_1, ..., a_n은 원자 객체의 비속성을 나타내고 있으며, 이들에 대한 정의역은 D이다. uta는 사용자 정의 시간이며, 이에 대한 정의역은 T이다.

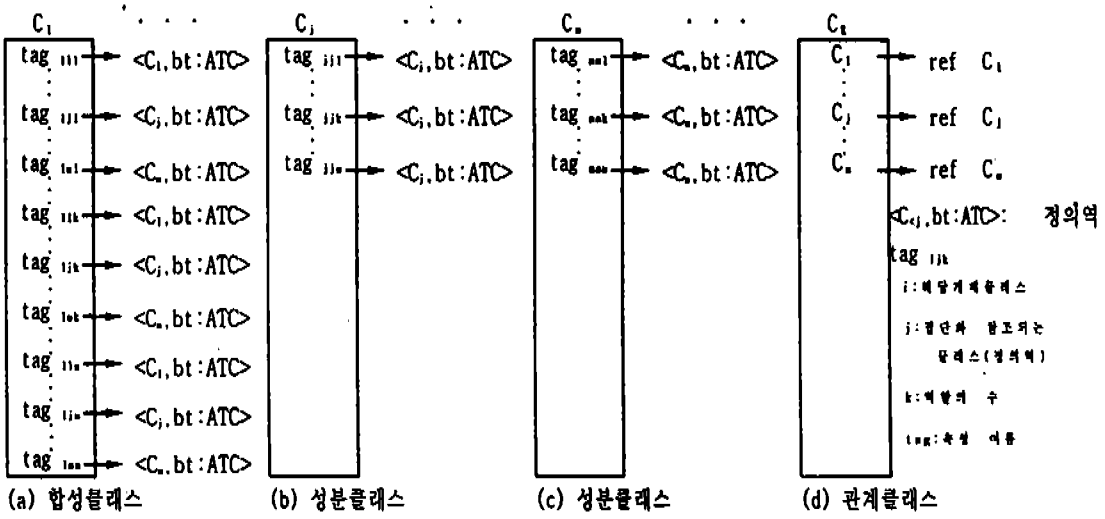
ta_1, ..., ta_n은 원자 객체의 시간 속성을 나타내고 있으며, 이들에 대한 정의역은 <D, bt: ATC>이다. 저장된 객체를 변경하거나 새롭게 객체를 생성하는 메소드들 $m_1(C)$, ..., $m_0(C)$ 에 대응하는 정의역은 반환하는 이력 객체의 여부와 합성 객체의 여부에 따라 D, <D, bt: ATC>, C, <C, bt: ATC>가 된다.

클래스 사이의 관계 의미가 집단화인 경우에 역할에 의한 클래스들을 연결하는 속성-영역 관계는 <그림 3.4>와 같이 표현할 수 있다.

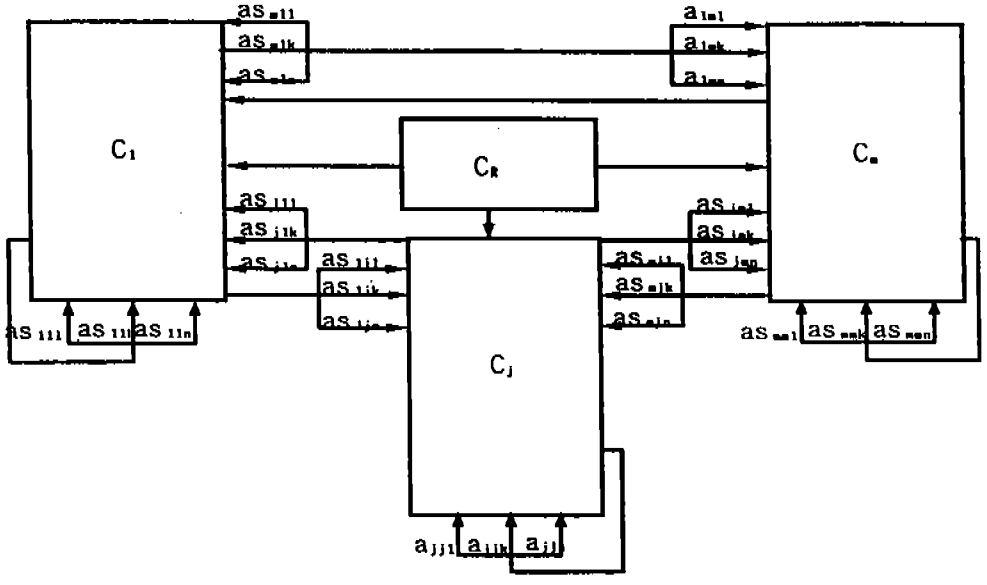
그림 3.4 집단화의 속성화에서 CE_1이 합성 클래스라 하고, 그 밖의 다른 클래스는 성분 클래스 $CE_j(1 \le j \le m)$라 하면, 집단화 의미에 의한 속성-영역 관계를 <그림 3.5>와 같이 나타낼 수 있다.



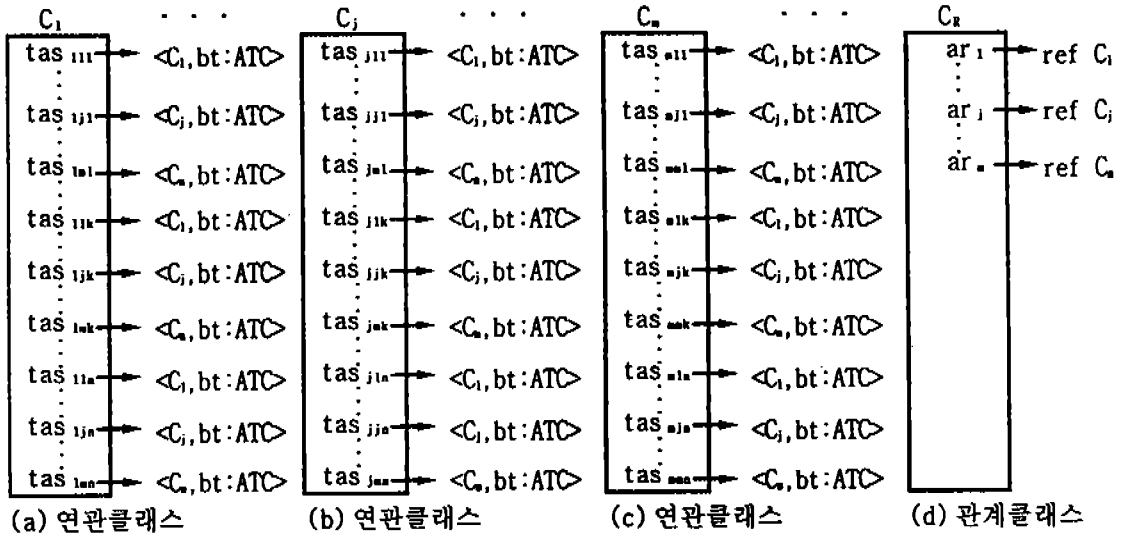
<그림 3.4> 속성 버전화에서 관계 의미를 가지는 집단화 속성화



<그림 3.5> 속성 버전화에서 역할에 의한 집단화의 속성-영역



<그림 3.6> 속성 버전화에서 관계 클래스를 가지는 연관화의 속성화



<그림 3.7> 속성 버전화에서 역할에 의한 연관화의 속성-영역

<그림 3.5> 에서 개체 클래스의 개수는 $m(1 \le i \le m)$이고, 해당 속성에 대한 정의역의 개수는 m개 $(1 \le j \le m)$이며, 역할의 개수는 n이다 $(1 \le k \le n)$.집단화 의미는 독립적, 반사, 반대적, 그리고 비순환적 특성에 의해서 임의의 한 합성 클래스를 C_1로 정하고, 그 밖의 클래스들은 클래스 C_2, ..., C_m으로 정하며, 자기 자신도 성분 클래스로 정한다. 그래서 속성 버전화에 의해서 비재귀적 참조에 의한 속성의 개수는 (m-1)n이며, 재귀적 참조에 의한 속성의 개수는 m×n이다. 전체 속성의 개수는 (2m-1)×n이다.

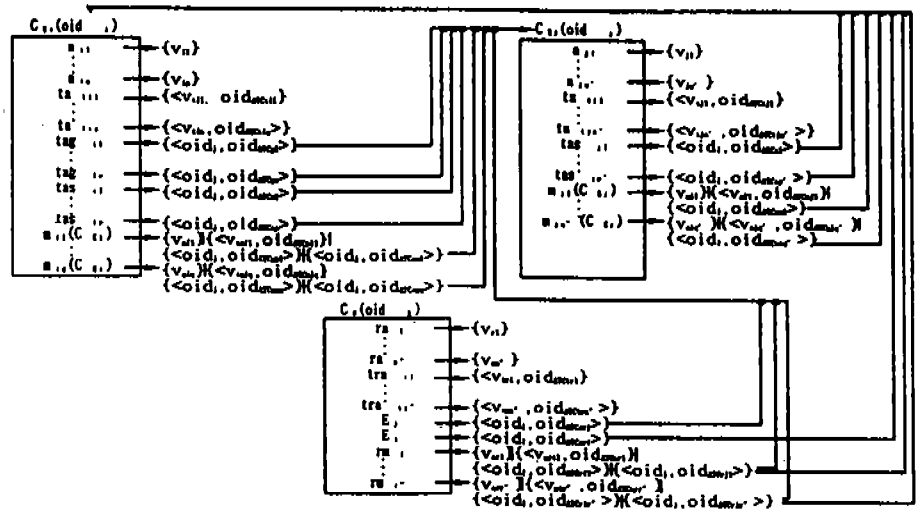
클래스 사이의 관계 의미가 연관화인 경우에 역할에 의한 속성-영역 관계는 <그림 3.6>과 같이 표현할 수 있다.

<그림 3.6>에서 임의의 한 클래스 $C_i(1 \le i \le m)$가 다른 클래스 $C_j(1 \le j \le m)$와 속성-영역 관계에 있는 클래스들을 규정할 수 있다.

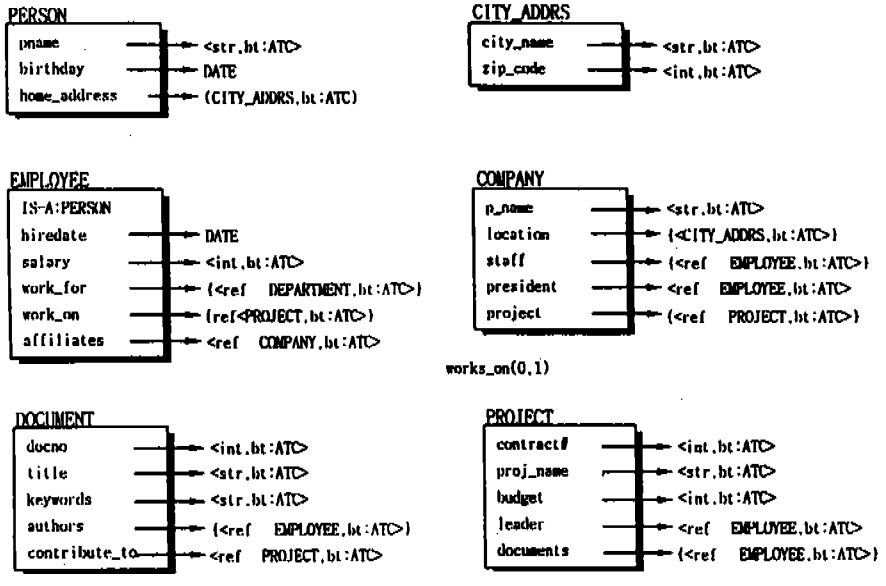
<그림 3.7> 연관화의 속성-영역에서 개체 클래스의 개수는 m이고, 정의역의 개수는 m이며, 영역의 개수는 n이다. 연관화는 종속적, 반사적, 생성적, 그리고 순환적 특성에 의해서 임의의 한 클래스 C_i를 고정하고, 이 클래스와 연관 관계가 있는 다른 클래스 C_j는 서로 재귀적이고 순환적 관계를 형성한다. 그래서 속성 버전화에 의한 연관화 속성 개수는 m×m×n이다.

속성-영역 관계에 있는 클래스들에서 생성된 객체를 관계 객체 따라 속성 버전화를 고찰하여 보자. 그래서 객체 입장에서 관계 객체가 존재하는 속성 버전화는 <그림 3.8>과 같다.

<그림3.8>에 표시된 관계 객체를 가지는 속성 버전화에서 C_{Ei}와 C_{Ej}는 관계 클래스들이고, C_R은 관계 클래스이다. $[a_{i1}, ..., a_{im}]$, $[a_{j1}, ..., a_{jm'}]$와 $[ra_1, ..., ra_{m''}]$은 비시간 속성들이며, $[ta_{ti1}, ..., ta_{tin}]$, $[ta_{tj1}, ..., ta_{tjn'}]$와 $[ta_{tr1}, ..., ta_{trn''}]$은 시간 속성들이다. $[tag_{i1}, ..., tag_{io}]$는 C_{Ei}를 합성 클래스로 하고, C_{Ej}를 성분 클래스로 했을 경우에 집단화 속성을 나타내고, 이 여러 개의 속성들은 집단화 객체들 사이의 역할을 의미한다. 그리고 집단화 속성은 역참조(함수)가 성립하지 않기 때문에 합성 클래스에만 존재한다.



<그림 3.8> 관계 객체를 가지는 속성 버전화



<그림 3.9> 속성 버전화를 위한 시간지원 객체지향 데이터베이스 스키마

$[tas_{i1}, ..., tas_{ip}]$와 $[tas_{j1}, ..., tas_{jp'}]$는 C_{Ei}와 C_{Ej}가 서로 연관화 참조를 할 경우에 속성들을 나타내고 이 여러 개의 속성들은 연관화 객체 사이에 역할을 의미 한다. 그리고 연관화 속성은 역참조(함수)가 성립하기 때문에 tas_i의 역참조 tas_j이며, 즉 $tas_i^{-1} = tas_j$이다.

$[m_{i1}(), ..., m_{iq}()]$ 이다. $[m_{j1}(), ..., m_{jq'}()]$와 $[rm_1() , ..., rm_{q''}()]$는 각각 C_{Ei}, C_{Ej}와C_R의 메소드들을 나타내며, 반환 값은 시간 값의 여부와 원자자료형의 여부에 따라 결정된다. $[a_{Ei}, a_{Ej}]$는 관계 객체가 개체 객체를 참조함을 나타낸다. 그래서 클래스 정의어 구문에 기반한 다이어그램에 의한 시간지원 객체지향 다이어그램은 다음과 같다.

3.4 속성 버전화의 시간지원 객체지향 데이터베이스 예

객체 사이의 관계 의미에 시간 속성은 클래스 계층 구조의 유무에 따라 속성-영역 관계를 규정할 수 있음을 앞에서 제안하였다. 세부적으로 한 클래스에 대한 표현 방법은 <그림 3.1>에 표현하였으며, 집단화 클래스 계층 구조에 의한 표현 방법은 <그림 3.4>와 <그림 3.5>에 기술하였으며, 연관화 클래스 계층 구조에 의한 표현 방법은 <그림 3.6>과 <그림 3.7>에 기술하였다. 2.3절의 객체지향 데이터베이스의 스키마의 예에 대응하는 시간지원 객체지향 다이어그램의 간단한 예는 다음 <그림 3.9>와 같다.

<그림 3.9>의 스키마에서 개체 클래스들은 CITY_ADDRS, COMPANY, PERSON, EMPLOYEE, PROJECT, DEPARTMENT 등이며, 개체 클래스 CITY_ADDRS에서 zipcode와 cityname은 다른 개체 클래스들과 관계하지 않는 시간 속성들을 나타내며, {home_address:<CITY_ADDRS, bt: ATC>}, total에서 home_address는 집단화 관계에 의한 속성이름이며, 이 속성의 비시간 객체의 정의역인 CITY_ADDRS는 관계하는 개체 클래스를 나타내며, bt는 유효시간 속성과 거래시간 속성과 사용자 정의 시간의 속성을 조합한 시간 속성이며, ATC는 시간 속성에 대한 추상화 클래스이다.

또한 total은 두 개체 클래스 CITY_ADDRS와 PERSON의 mull 값을 가지지 않음을 나타내며, staff: {<ref EMPLOYEE, bt: ATC>}, total에서 staff는 연관화 관계에 의한 속성 이름이며, 이 속성의 객체의 정의역 EMPLOYEE는 참조되는 개체 클래스를 나타내며, bt는 유효시간 속성과 거래시간 속성과 사용자 정의 시간의 속성을 조합한 시간 속성이며, ATC는 시간 속성에 대한 정의역 이다.

4. 제약 조건

제 4.1절에서는 속성 버전화에 대한 제약조건을 기술한다. 제 4.2절에서는 객체 표현 기준에 대해 분석한다.

4.1 속성 버전화의 제약 조건

이 시간지원 속성 함수에 대한 제약조건을 규정하기 위해서는 생명주기함수lifespan(), 시간지원 객체 o의 시간지원 속성 함수 $f_i(o)$이며, 시점 t가 필요하다. 그래서 이에 대한 제약조건은 다음과 같이 기술할 수 있다[Wuu92]. 객체 o, 클래스 C, 그리고 데이터베이스 DB 사이에 생명주기가 존재한다. 객체 o가 데이터베이스 DB에서 클래스 C에 있는 것이라 하면, 다음의 조건이 성립 한다.

(C_1) $lifespan(o/C) \subseteq lifespan(C) \subseteq lifespan(DB)$

시간 객체 o의 생명주기와 그 시간 속성 f_i에 대해서 다음의 제약 조건이 성립한다.

(C_2) $|f_i(o)| \subseteq lifespan(o)$

예로 한 고용인이 시간 t 동안에 한 부서에서

근무했다면, 다음이 성립한다.

$t \in |department(e)| \subseteq lifespan(e)$

일반화 클래스 계층에서 o가 부분 클래스 S와 상위 클래스 P 둘 다의 객체라 하면, 다음의 제약조건이 성립한다.

(C_3) $lifespan(o/S) \subseteq lifespan(o/P)$

집단화 클래스 계층에서 합성 클래스 C_i의 객체 oi에 대한 생명주기와 성분 클래스 C_j의 객체 oj에 대한 생명주기는 다음과 같다.

(C_4) $lifespan(o_j/C_j) \subseteq lifespan(o_i/C_i)$

연관화 클래스 계층에서 독립적인 두 클래스 C_i 와 C_j에 대한 생명주기의 제약조건은 한 클래스 C_i 의 시간 속성 fi와 정의역 C_j의 관계 1:1, 1:N 그리고 M:N에 의해서 규정된다. 이에 대해서 시간지원 속성에 부과되는 유일한 제약은 그 버전 리스트에서 두 쌍 (v_1, t_1)와 (v_2, t_2)에 대해서 $t_1 \cap t_2 = \varnothing$이 된다는 것이다.

관련된 객체들의 참조속성들 사이에 유지되어야 하는 제약 조건은 아래에 주어져 있다. 그리고 다음의 c_1에서 참조-리스트$_1$과 c_2에서 참조-리스트$_2$는 같은 관계성을 나타내는 역 시간지원 참조 속성이다. 이 제약 조건들은 두 시간지원 참조 속성에서 역 특성을 규정한다. 그래서 연관화에 대한 세부적인 제약조건은 다음과 같다.

1. C_i 과 C_j 사이에 1:1 관계일 때의 시간 제약 조건은 다음과 같다.

(C_5) 참조-리스트$_1$에 있는 모든 시간 값 (v_i, t_i)에 대해서, $t_i = t_j$ 이도록 하는 참조-리스트$_2$에 적어도 하나의 (vj, tj)의 값이 존재한다.

(C_6) 참조-리스트$_2$에 있는 모든 시간 값 (v_i, t_i)에 대해서, $t_i = t_j$ 이도록 하는 참조-리스트$_1$에 적어도 하나의 (v_j, t_j)의 값이 존재한다.

2. C_i 과 C_j 사이에 1:N 관계일 때의 시간 제약 조건은 다음과 같다

(C_7) 참조-리스트$_1$에 있는 모든 시간 값 (v_i, t_i)에 대해서, $t_i \subseteq t_j$ 이도록 하는 참조-리스트$_2$에 적어도 하나의 (vj, tj)의 값이 존재한다.

(C_8) 참조-리스트$_2$에 있는 모든 시간 값 (v_i, t_i)에 대해서, $t_i = \bigcup_{j=1}^{N} tj$ 이도록 하는 참조-리스트$_1$에 적어도 하나의 (v_j, t_j)의 값이 존재한다.

3. C_i 과 C_j 사이에 M:N 관계일 때의 시간 제약 조건은 다음과 같다.

(C_9) 참조-리스트$_1$에 있는 모든 시간 값 (v_i, t_i)에 대해서, $t_i = \bigcup_{j=1}^{N} tj$ 이도록 하는 참조-리스트$_2$에 적어도 하나의 (vj, tj)의 값이 존재한다.

(C_{10}) 참조-리스트$_2$에 있는 모든 시간 값 (v_i, t_i)에 대해서, $t_i = \bigcup_{j=1}^{N} tj$ 이도록 하는 참조-리스트$_1$에 적어도 하나의 (v_j, t_j)의 값이 존재한다.

한 클래스의 시간 외연에 대한 생명주기 t_extent(C)(t)와 한 클래스의 객체에 대한 생명주기 lifespan(o/C)의 관계는 다음과 같은 제약조건을 가진다.

(C_{11}) $o \ni t\_extent(C)(t) \Leftrightarrow t \in lifespan(o/C)$

일반화 클래스 계층에서 시점 t를 가지는 부

분 클래스 S와 상위 클래스 P의 쌍에 대해서 (C3)와 (C_{11})로부터 다음의 제약조건이 성립한다.

(C_{12}) t_extent(S)(t) ⊆ t_extent(P)(t)

집단화 클래스 계층에서 시점 t를 가지는 합성 클래스 C_i와 성분 클래스 C_j의 쌍에 대해서 (C_4)와 (C_{11})로부터 다음의 제약조건이 성립된다.

(C_{13}) t_extent(C_j)(t) ⊆ t_extent(C_i)(t)

4.2 검토

이 절에서는 객체 표현을 위한 세 가지 기준을 고려한다. 그리고 각 기준에 따른 표현 방법을 검토한다.

기준 1. 규정된 시간지원 데이터베이스를 모델링하기 위해서 필요한 추가적인 클래스의 수

속성 버전화에 의한 객체지향 모델은 이력 사항을 요구하는 속성에 추상화 시간 클래스를 추가한다. 그래서 속성 버전화 필요한 클래스의 수는 $\#E + \#R + q \times \#ATC$ 가 되며, 여기서 q는 이력 사항을 요하는 속성의 수를 나타낸다. 객체 버전화에 의한 객체지향 모델은 개체 클래스, 관계 클래스, 그리고 이 개체와 관계의 클래스마다 이력 사항을 요구하는 추상화 시간 클래스가 포함된다. 그래서 객체 버전화에 필요한 클래스의 수는 $\#E + \#R + 2 \times \#ATC$ 가 된다. 결국 관계하는 클래스의 수가 많을수록 객체 버전화가 클래스의 수가 적다.

기준 2. 실세계 객체를 표현하기 위한 데이터베이스의 수

각 실세계 사건은 속성 버전화에 대한 데이터베이스의 객체 사이에 1 : 1 대응한다. 그러나 개체의 속성에 값들이 데이터베이스화했을 때 일반화와 집단화의 상속성에 의해 감소한다.

기준 3. 정보의 시간지원 단위의 granularity

속성 버전 모델은 각 클래스의 시간지원 속성에 이원 시간을 추상 클래스화하여 사용하고 있으나, 객체 버전화는 이원 시간을 일률적으로 각 개체 클래스와 관계 클래스에 단 한 번만 포함시켜 사용한다.

결국 속성 버전화 모델은 시간 값과 비시간 값을 동시에 표현하며, 합성 함수 기법을 이용하여 속성 영역 관계를 형성한다. 집단화 참조에 의한 속성 영역 관계는 정의역을 성분클래스와 추상화 시간 클래스를 조합하여 표현하며, 연관화 참조에 의한 속성-영역 관계는 정의역을 연관화된 클래스와 추상화 시간 클래스를 조합하여 표현한다.

5. 결 론

시간지원 객체지향 데이터베이스는 시간의 흐름에 따라 변화된 객체를 조작할 수 있는 데이터베이스이다. 기존의 스냅샷 데이터베이스는 이력자료를 취급해야 하는 CAD/CAM, 통계적 데이터 처리, 법령의 저장, 감사 추적, 범죄 추적, 그리고 의료 분야 등과 같은 분야에서 데이터의 변화 추이를 관찰하기가 어렵다. 또한 데이터 웨어하우징에서의 이력 관리에도 효과적으로 이용된다. 따라서 이와 같은 응용분야의 문제를 해결하기 위하여 시간지원 객체지향 데이터베이스 시스템에서는 실세계의 복잡한 이력 객체들을 더욱 강력하게 표현하고, 간단하게 조작할 수 있도록 한다.

관계형 데이터 모델에서는 제일 정규형이나 비제일정규형에 의해서 영향을 받지만, 객체지향 모델은 복합 객체를 위한 강력한 구성자를 지원하기 때문에 시간지원 객체지향 데이터베이스는

정규화에 영향을 받지 않는다. 그러나 기존의 객체지향 모델은 합성 객체와 성분 객체 사이를 연결하는 집단화 참조와 임의의 객체들 사이를 연결하는 연관화 참조는 의미와 구문을 나누지 않았다. 이것은 객체의 합성과 객체의 임의의 연관화 사이에 애매모호성을 증가시키며, 합성 객체와 성분 객체 사이에 구조와 행위의 상호 작용에 대해서 오류를 발생시킬수 있다.

따라서 이 논문은 시간 개념이 포함된 객체지향 모델에서 관계 의미를 일반화, 집단화와 연관화로 구분하여 형식적으로 표현하는 방법을 제시하였다. 이 연구는 유도형 시간 속성을 추상화 클래스화시켜 속성-영역에서 시간 영역에 도입하였고, 일반화, 세부화, 집단화와 연관화의 의미에 따라 메소드 및 속성에 속성 버전화에 유효시간과 거래시간을 클래스 다이어그램과 클래스 정의어 구문을 규정하였다. 이 연구는 실세계에서 이력 사건들 사이의 관계를 일반화, 집단화와 연과화로 구분하므로서 이 관계들의 역할을 객체지향 모델의 속성-영역 관계에 도입하여 질의처리를 구현하는데 적용할 수 있고, 다형성 관점에서 일반화 상속성과 집단화 상속성을 규정함으로써 재사용성을 증가시킬 수 있으며, 이 시간지원 객체지향 모델을 형식화함으로서 질의의 해석식과 대수 연산자를 규정할 때 강력한 연산기능을 제공할 수 있다.

이 연구에서 제안한 변환 방법은 기존의 데이터베이스 시스템에 적용 및 새로운 시스템의 설계와 더불어 그 적용을 연구 중에 있으며, 아울러 제시한 모델에 기반하여 시간지원 객체지향 질의 조작 구문을 설계하고, 이에 기반한 질의 처리기를 구현하려고 한다.

〈참고문헌〉

[Ame90] America P., "A behavioral approach to subtyping in object-oriented programming languages," Proc. Foundations of Object-oriented languages, 1990.

[Ber91] Bertino, E., and Marrino, L., "Object-oriented database management systems: concepts and issues," IEEE Computer, 1991.

[Car85] Cardelli, L. and Wegner, P., "On Understanding types, Data Abstraction, and Polymorphism," ACM Computer Surv., Vol.17, No.4, December 1985, pp.471-522.

[Che76] Chen P.P., "The Entity-relationship Model-Toward a Unified View of Data," ACM TODS, Vol.1, No.1 May 1976, pp.9-35.

[Deu91] Deux, O., et al., *The O_2 system*, ACM, October 1991.

[Elm90a] Elmasri, R., El-Assal, I., Kouramajian, V., "Semantics of temporal data in an extedned ER model," In 9th Entity-relationship Conference, 1990.

[Elm90b] Elmasri, R., and Wuu, G., "A temporal model and query language for ER database," In IEEE Data Engineering Conference, 1990.

[Elm93b] Elmasri, R., Kouramajian, V. and Fernado, S., "Temporal Database Modeling: An Object Oriented Approach," In international Conference on Information and Knowledge Management,, November 1993, pp.574-585.

[Elm93a] Elmasri, R., Kouramajian, V. and Fernando, S, "Temporal Database Modeling: An Object Oriented Approach," In international Conference on Information and Knowledge Management,, November 1993, pp.574-585

[Elm93b] Elmasri, R., and Navathe, S., *Fundamentals of Database systems*, 2nd Ed. Benjamin /Cummings, 1993.

[Gad88a] Gadia, S. K., "A homogeneous relational model and query languages for temporal databases," ACM Trans. Database Syst., Vol.13, No.4, 1988, pp.418-448.

[Gad88b] Gadia, S., and Yeung, C., "A generalized model for a temporal relational database," In ACM SIGMOD Conference, 1988.

[Jen90] Jense, C., "Towards the realization of transaction time database system," Ph.D Dissertation, University of Maryland, 1990.

[Kim91] Kim, W., *Introduction to Object-oriented databases*, The MIT Press, 1991.

[Lec88] Lécuse, C., Richard, P., and Velvez, F., "O_2, an object-oriented data model," In ACM SIGMOD Conference, 1988.

[Lee94] Lee, H.R., "A Logical Optimization of Queries in Object Oriented Database System," PhD Thesis, the Chonbuk National Univ. at Chonbuk in Korea 1994.

[Lee95] Lee, H.R, et al., "Logical Optimization of Queries using a Complex Object Calculus Transformation," Journal of KISS(B): Software and Applications, Vol.22, No.12, December 1995, pp.160-1613.

[Lie93] Lieberherr K. J. and Xiao C., "Formal foundations for object-oriented data modeling," IEEE Transactions on Knowledge and Data Engineering, Vol.5, No.3, June 1993, pp.462-478.

[Lin93] Ling L,. "A Recursive object algebra based on aggregation for manipulating complex objects," Data & Knowledge engineering, Vol.11, North-Holland, 1993, pp.21-60.

[Nav89] Navathe, s., and Ahmed, R., "A temporal data model and query language," In Information Science, Vol.49, No.2, pp.147-175, 1989.

[Neu89] Neuhold E, et.al, "Separating structural and semantic elements in object-oriented knowledge bases," Proc. Advanced Database Sys. Kyoto, 1989.

[Pis92] Pissino, N., and Makki, K., "T-3DIS: an approach to temporal object databases," In international Conference on Information and Knowledge Management,. 1992. pp.176-185.

[Sno87] Snodgrass, R., "The temporal query language TQUEL," In ACM TODS Vol. 12, No.2, 1987.

[Sno94] R. Snodgrass, Nick Kline, "Aggregate in TSQL2," The TSQL2 Language Design Committee, Mar. 21, 1994.

[Tan86] Tansel, A. U., "Adding time dimension to relational model and extending relational algebra," Inf. Syst., Vol.11, No.4, pp.343-355, 1986.

[Tuz90] Tuzhilin, A. and Clifford, J., "A temporal relational algebra as a basis for temporal relational completeness," In International Conference on VLDB, pp.13-23, 1990.

[Wuu92] Wuu, G., and Dayal, U., "A Uniform model for temporal object-oriented databases," In IEEE Data Engineering Conference, 1992.

[정경자96] 정경자, 김동호, 이언배, 류근호, "시간지원 데이터베이스의 질의 처리," 데이터베이스 연구회지, 12권, 3호, 1996, 8.

[김삼남97a] 김삼남, 이홍로, 류근호, "개체-관계성 모델을 관계의 역할에 따른 객체지향 데이터베이스 모델로 변환," 정보처리학회지, 4권, 7호, 1997.

[김삼남97b] 김삼남, 이홍로, 류근호, "관계 의미에 따른 개체-관계 데이터 모델의 객체-지향 데이터 모델로 변환," 정보처리과학논문지(B), 24권, 제?호(6월 26일 심사완료), 1997.