차 ● 례

나 민 영 / University of Florida 전산학박사
육군사관학교 전산학교수

전문가리포트

데이터베이스 설계(III)

3. 분산 데이터베이스 설계

분산 데이터베이스 시스템은 일반적으로 신뢰도와 가용도를 높이며 실제로 우리 주변에서 데이터베이스의 응용이 대부분 분산적이기 때문에 매우 중요하게 취급되어 연구가 한창 진행중인 분야이다. 분산 데이터베이스 설계를 위한 방법으로서는 Top-down 방법과 Bottom-up 방법의 두 종류가 있다. Top-down 방법은 데이터베이스를 나누어 퍼뜨리는 전형적인 방법이고, Bottom-up 방법은 이미 존재하는 데이터베이스들을 묶어 통합된 멀티 데이터베이스를 만들어내는 전형적인 방법이다. 본 장에서는 Top-down 방법에 의한 설계를 다룬다. Top-down 방법에서는 전역스키마가 주어지고 데이터베이스의 분할이 설계된후 이 파편들이 각 장소로 할당되어진다. 그림 9는 분산데이터베이스 설계가 Top-

down 방법으로 수행될때 그 개요를 보여주고 있다. 중앙집중형 데이터베이스와 비교할때 큰 차이점은 분산설계 (Distribution Design) 단계이다.

분산설계란 분산 데이터베이스 구현시 가장 기본이 되는 단계로서 주어진 트랜잭션 요구사항에 대하여 데이터베이스를 컴퓨터 네트워크 상에 최적으로 분산시키는 문제를말한다.

데이터를 분산설계하는 목적은 데이터를 사용

자에게 가능한한 가까이 위치시켜 지역적 처리의 극대화를 꾀하고, 분산된 데이터의 가용성(Availability)과 신뢰성(Reliability)을 증가시키며, 응용 프로그램의 병렬적 수행으로 부하를 분산시키고자하는 것이다. 분산설계를 위한 방안으로서는 분할, 할당, 및 중복이 있는데 분할은 수직분할, 수평분할, 혼합분할로 나뉘어진다.

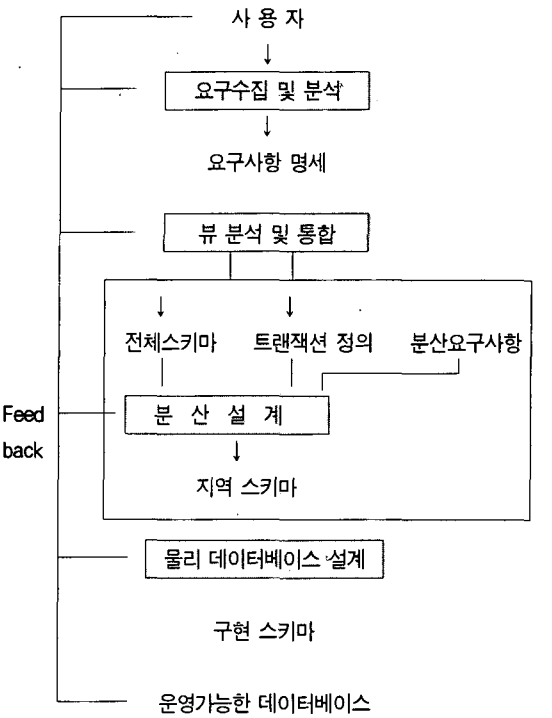

그림 9. 분산 데이터베이스 설계과정 (Top-down 방법)

수직분할 (Vertical Fragmentation)

수직분할이란 릴레이션과 같은 전역 객체를 애트리뷰트의 그룹으로 나누는 작업으로서 트랜잭션 명세, 애트리뷰트 사용행렬, 애트리뷰트 친화도 행렬 등이 준비되어야 한다. 애트리뷰트 사용행렬은 주요 트랜잭션에 의한 애트리뷰트의 사용을 나타내는 행렬로서 한 행은 하나의 트랜잭션에 관한 정보를 나타내는데 그 원소값이 1이면 애트리뷰트가 트랜잭션에 의해 사용됨을 나타내고 값이 0이면 그렇지 않음을 나타낸다. 여기서 주요 트랜잭션이란 80/20 규칙에 따르는 트랜잭션을 의미하는 것으로 이는 데이터베이스 액세스의 80%를 차지하는 사용자들이 20%의 주요 트랜잭션을 사용함을 의미한다.

애트리뷰트 사용행렬 작성 예

애트리뷰트 사용행렬을 작성하기 위해 먼저 주요 트랜잭션을 기술하면 다음과같다.

T1 : Find the budget of a project, given its identification number

```
SELECT  BUDGET
FROM    J
WHERE   JNO=Value
```

T2 : Find the numbers, names and bugets of all projects

```
SELECT  JNO, JANME, BUDGET
FROM    J
```

T3 : Find the names of projects located at a given city

```
SELECT  JNAME
FROM    J
WHERE   LOC=Value
```

T4 : Find the total project budgets for each city

```
SELECT  SUM(BUDGET)
FROM    J
WHERE   LOC=Value
```

이때 A1=JNO, A2=JNAME, A3= BUDGET, A4=LOC 라 하면 애트리뷰트 사용행렬은 다음과 같이 작성된다. 이 행렬에서 트랜잭션 T1은 애트리뷰트 A1과 A3를 사용함을 나타내고 있음을 알수 있다.

	A1	A2	A3	A4
T1	1	0	1	0
T2	1	1	1	0
T3	0	1	0	1
T4	0	0	1	1

애트리뷰트 친화도 행렬

애트리뷰트 친화도(Attribute Affinity : AA)란 두 애트리뷰트간 결속력을 측정하는데 사용되는 값으로

$$AFFij = \sum_{k} ACCkij$$

와 같이 정의된다. 여기서 ACCkij 는 두 애트리뷰트 i와 j를 모두 참조하는 트랜잭션K의 액세스 이때는 트랜잭션의 셑이다.

이제 앞에서 작성된 애트리뷰트 사용 행렬로부터 애트리뷰트 친화도 행렬을 구해보자. 먼저 ACC1=45, ACC2=5, ACC3=75, ACC4=3 이라고 가정하면 다음과 같은 친화도 행렬을 얻을 수 있다. 여기서 A(1,3)의 값 50은 애트리뷰트 A1과 A3 를 모두 사용하는 트랜잭션 T1과 T2의 액세스의 수 45와 5를 합하여 얻어진 것이다.

	A1	A2	A3	A4
A1	50	5	50	0
A2	5	80	5	75
A3	50	5	53	3
A4	0	75	3	78

이와같이 애트리뷰트 친화도 행렬이 구성되면 적절한 알고리즘(목적함수나 그래프이용)을 사용하여 이 행렬을 클러스터링한 후 애트리뷰트의 그룹으로 나누면 수직파편이 생성되어 수직분할이 완료된다. 앞의 예는 다음과 같이 클러스터링되어 A1과A3가 하나의 수직파편으로 되고 A2와 A4가 다른 하나의 수직파편으로 되어 수직분할이 이루어진다.

	A1	A3	A2	A4
A1	50	50	5	0
A2	50	53	3	3
A3	5	5	80	75
A4	0	3	75	78

수평분할 (Horizontal Fragmentation)

수평분할은 전역 릴레이션을 수평파편이라 불리는 투플들의 셑으로 쪼개는 작업으로 그 종류에는 primary 분할과 derived 분할이 있다. 이 중 primary 분할은 우리가 보통 생각하는 분할로서 전역 릴레이션 상에서 selection 연산자를 사용하여 정의되는데 반해 derived 분할은 릴레이션 자신의 애트리뷰트의 성질에 기인하는 것이 아니라 다른 릴레이션의 수평분할로부터 유도되는 수평분할로서 파편간 조인을 쉽게 하기 위해 사용된다. 예를 들어 그림 10(a)와 같은 릴레이션 J,G가 있을때

J1 = LOC="Montreal"(J)
J2 = LOC="New York"(J)
J3 = LOC="Paris"(J)

과 같은 selection 연산자에 의한 릴레이션 J의 primary 분할은 그림 10(b)와 같게 된다. 이때 만일 어떤 장소에서 일하는 엔지니어들의 이름을 검색하는 응용 프로그램이있어서 릴레이션 J와 G를 JNO 애트리뷰트 상에서 자주 조인하게 된다고 하면 릴레이션J가 이미 JNO값에 따라 J1, J2, J3로 수평분할 되어 있으므로 릴레이션 G 역시 그림10(c)와 같은 3개의 유도된 수평파편으로 분할이 이루어질 수 있다. 이러한 수평분할을 위한 기법에는 민텀 프레디키트 기법, adaptive 클러스터링 기법, 그래프를 이용한 기법 등이 있다.

J

JNO	JNAME	BUDGET	LOC
J1	Instrumentation	15,000	Montreal
J2	Database Develop	135,00	New York
J3	CAD/CAM	250,000	New York
J4	Maintenance	310,000	Paris

J1

JNO	JNAME	BUDGET	LOC
J1	Instrumentation	150,000	Montreal

J2

JNO	JNAME	BUDGET	LOC
J2	Database Develop	135,000	New York
J3	CAD/CAM	250,000	New York

J3

JNO	JNAME	BUDGET	LOC
J4	Maintenance	310,000	Paris

G

ENO	JNO	RESP	DUR
E1	J1	Mager	12
E2	J1	Analyst	24
E2	J2	Analyst	6
E3	J3	Consultion	10
E4	J4	Engineer	48
E4	J2	Programmer	18
E5	J2	Manager	24
E6	J4	Manager	48
E7	J3	Engineer	36
E8	J3	Manager	40

G1

ENO	JNO	RESP	DUR
E1	J1	Manager	12
E2	J1	Analyst	24

G2

ENO	JNO	RESP	DUR
E2	J2	Analyst	6
E3	J3	Consultant	10
E4	J2	Programmer	18
E5	J2	Manager	24
E6	J3	Engineer	48
E7	J3	Manager	36

G3

ENO	JNO	RESP	DUR
E3	J4	Engineer	48
E6	J4	Manager	48

(a) 보기 릴레이션 (b) Primary 분할

(c)derived 분할

그림 10. 수평분할

혼합분할 (Mixed Fragmentation)

혼합분할은 수직분할과 수평분할의 혼합형으로서, 이에 관한 연구는 아직 그리 많지 않은 실정이다. 따라서 보통 혼합분할을 위하여서는 한 릴레이션을 수평적으로 또는 수직적으로 쪼개는 방법을 쓰고 있는데 이는 수평분할 후 수직분할 또는 수직분할 후 수평분할의 두가지 방법으로 이루어진다. 그러나 이러한 방법은 분할의 순서에 따라 그 결과가 달라질 수 있으므로 적합치 못하다. 따라서 최근에는 수평분할과 수직분할을 동시에 적용하여 하나의 릴레이션을 격자방 (grid cells)들로 나눈후 이 방들을 트랜잭션 처리비용이 최소화되도록 합병하여 수직분할이나 수평분할로는 얻을수 없는 혼합파편 (mixed fragments)이라 불리는 분할결과를 만들어 내는 기법을 사용한다.

할당 (Allocation)

할당이란 파편의 셑 F = { F1, F2, ..., Fn } 와 사이트 S = { S1, S2, ...,Sm }로 구성된 네트워크 상에서 응용의 세트 Q = { q1, q2, ..., qn }가 돌아가고 있을때 F를 S로 최적으로 분산하는 문제를 말한다. 여기서 최적화 (Optimality)란 저장비용, 질의비용, 통신비용 등 비용의 최소화 (Minimal Cost)와 응답시간, 시스템 산출물, 메모리 사용등 성능(Perfor-mance) 향상의 두 측면을 고려한 것이어야 한다.

할당은 통상 중복과 병행해서 생각할 수 있으므로 모든 데이터베이스가 모든 사이트에 중복되어 할당되는 경우 (Fully-replicated), 데이터베이스의 일부 파편이 중복되어 할당되고 일부 파편은 그렇지 않은 경우 (Partially-replicated), 하나의 사이트에 오직 하나의 파편만 할당되어 저장되는 경우 (Non-replicted) 등으로 나뉜다.

이러한 데이터베이스의 할당은 한 파편의 할당이 다른 파편의 할당에 영향을미칠수 있고, 분산 데이터베이스시스템에 있어서 데이터 액세스는 화일 시스템에서의 액세스보다 훨씬 더 복잡하므로 기존의 화일 할당 (FAP : File Allocation Problem)보다 훨씬 더 어려운 작업이라 할 수 있다. 효과적인 데이터베이스 할당을 위해서는 다음과 같은 정보들이 요구되어진다.

- Database 정보
  - 분할을 위한 정보
  - 카디날리티
  - 애트리뷰트 길이
  - 할당의 단위
- Application 정보
  - 검색 액세스 정보
  - 갱신 액세스 정보
  - 액세스 빈도
  - 응답시간 제약 조건
- Site 정보
  - 저장장치 용량
  - 프로세싱 능력
  - 사이트의 수
- Network 정보
  - 데이터 통신 단위 비용
  - 프로토콜 오버헤드

4. 연합 데이터베이스 설계

우리 주변에는 서로 이질적인 여러 데이터베이스들이 많이 존재하고 있다. 그이유는 데이

터베이스들이 이미 개발되어 사용되고 있으며 현재 대부분 업무의 기능 및 응용 성격이 분산적이기 때문이다. 이질성의 대부분 형태는 기술적 차이에서 비롯된다. 예를들면 하드웨어, 소프트웨어, 그리고 통신 시스템 같은 것들이다. 데이터베이스 시스템에 있어서 이질성은 구조의 차이, 제약조건에서의 차이, 질의어에서의 차이등과 같은 DBMS의 차이와 데이터의 시맨틱에 있어서의 차이에 기인한다.

이러한 이질적인 데이터베이스들을 연합(통합)하게 되면 데이터의 공유를 허용할 수 있고 응용 프로그램의 개발을 쉽게 할 수 있으며 또한 관련된 데이터의 일치를 꾀하며 중복을 피할수 있게 되므로 이 분야에 관한 연구는 매우 중요하게 인식되고있다. 연합의 개념은 우리의 실생활에서 많이 볼 수 있다. 예를 들어 국제연합(UN)을 생각해보자. UN은 그 구성원이 서로 다른 체제와 자치성을 갖는 이질적인 나라들로 구성되어 있으면서도 국제적인 이슈에 대해서는 권고하고 더 나아가 힘까지도 행사한다.

이와 같이 데이터베이스 연합의 가장 중요한 특징은 독립적인 시스템들의 협동인 것이다. 따라서 연합 데이터베이스 시스템은 서로 협동하는 그러나 자치적인 데이터베이스시스템들로 구성된다.

연합 데이터베이스(Federated Databases)의 개념은 1979년 Hammer와 Mcleod에 의해 최초로 제안되고 1982년 Heimbigner에 의해 확장된 이후로 최근 여러가지 모양으로 활발히 연구가 이루어지고 있다. 그 한가지 접근방법은 전역스키마(통합스키마, global schema, universal schema)를 사용하는 것이고 다른 접근방법은 전역 스키마를 사용하지 않는 것이다. 전역스키마를 사용하는 시스템에서는 스키마 변환 및 스키마 통합을 통해 전역스키마를 구성한다음 이 전역스키마에 해당되는 데이터 조작언어를 이용하여 질의어를 표현한다. 이러한 연구의 대표적인 예로서는 CCA의 MULTIBASE, UNISYS의 Mermaid 등이다. 전역스키마를 사용하지 않는 시스템에서는 스키마 통합을 하지 않는 대신 데이터베이스 조작언어에 여러가지 기능을 추가하여 강력한 데이터베이스 언어를 정의한 다음 이 언어를 이용하여 여러 데이터베이스로부터 정보를 추출해낸다. 대표적인 예로서는 Litwin 등의 Multidatabase System, 휴스톤 대학의 Omnibase 등이 있다. 이와 같은 강력한 데이터베이스 언어를 이용하는 시스템들은언어 번역뿐만 아니라 이질적인 목표스키마를 사용자 자신의 스키마로 뷰(view)함을 허용하는 스키마변환을 통해서 더 강력한 시스템으로 발전할 수 있다.

연합 데이터베이스 구조를 설명하기 위해서는 중앙집중형 DBMS에서의 3단계스키마 구조는 적합하지 못하다. 따라서 3단계 스키마 구조를 연합 데이터베이스 시스템(FDBS)에 맞도록 확장해야 한다. 참고로 Sheth가 제안한 5단계 구조는 다음과 같다.

(1) 지역 스키마 (Local Schema)

지역 스키마는 참여 데이터베이스 시스템의 개념적 스키마로서 참여 DBMS의 데이터 모델로서 표현된다. 따라서 서로 다른 지역스키마는 서로 다른 모델로서 표현된다.

(2) 구성스키마 (Component Schema)

구성스키마는 지역스키마를 FDBS의 캐노니칼 혹은 공통 데이터모델(CDM:Common Data Model) 이라 불리는 데이터 모델로 변환함으로서 얻어진다. 구성스키마를 CDM으로 정의하는 이유는 첫째, 하나의 표현방법을 사용해서 서로 다른 모든 지역스키마를 나타낼 수 있고, 둘째, 지역스키마에서 잃어버린 의미가 구성스

키마에서 되살아날 수 있기 때문이다.

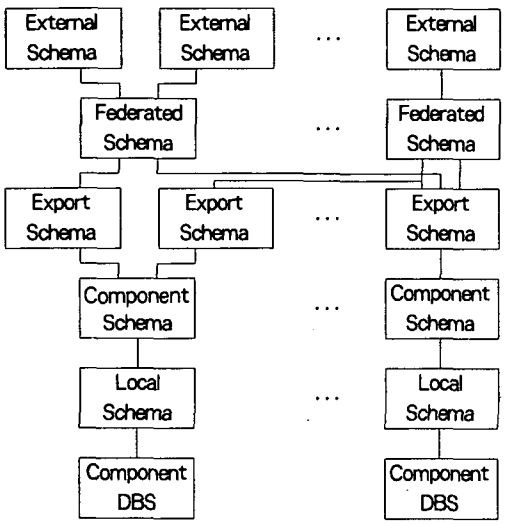

그림 11. 5단계 스키마 구조

(3) 수출 스키마 (Export Schema)

구성 DBS의 모든 데이터가 연합 사용자에게 가용한 것은 아니다. 수출스키마는 구성 스키마 중 FDBS에 가용한 서브셋을 나타낸다. 수출스키마를 정의하는 목적은 연합자치의 제어 및 관리를 쉽게 하기 위해서이다.

(4) 연합스키마 (Federated Schema)

연합스키마는 여러개의 수출스키마가 통합된 것으로 수출스키마를 통합할때 발생된 데이터 분산에 관한 정보를 포함한다.

(5) 외부스키마 (External Schema)

외부스키마는 사용자, 응용프로그램, 또는 사용자 클래스를 위한 스키마이다.

이상에서 살펴본 바와 같이 연합 데이터베이스를 구성하기 위한 스키마는 스키마변환과 스키마 통합을 통해 얻어진다.

스키마변환 (Schema Translation)

스키마 변환은 입력스키마로부터 출력스키마를 얻는 작업이다. 스키마변환에는 minimality를 위한 변환, 표현성 증가를 위한 변환, 정규화를 위한 변환, 이질 모델 액세싱을 위한 변환 등이 있으나 여기서의 스키마변환이란 이질 모델 액세싱을 위한 변환을 말한다. 이는 즉, 한 사용자가 다른 모델의 데이터베이스를 액세스하려고 할때에는 원시스키마로부터 목표스키마로 스키마변환이 이루어져야 함을 의미한다. 예를 들어 CDM이 EER 모델이고 연합에 참여하는 DBMS의 하나는 관계형이고 다른 하나는 CODASYL이며 연합사용자들이 관계형 뷰를 원한다고 하자. 그러면 (1) 관계형 스키마의지역스키마를 EER로 표현된 동등한 구성스키마로 변환하는 작업, (2) EER 모델의 연합스키마의 일부를 관계형 모델로 표현된 외부스키마로 변환하는 작업 등이 요구된다.

스키마변환을 위한 가장 기본적인 접근방법은 원시스키마로부터 목표스키마를 생성해내기 위한 매핑규칙을 개발하는 것이다. 이러한 규칙들은 목표스키마내의 각 객체들이 어떻게 원시스키마내의 객체로부터 유도되는가를 명시한다. 만일 매핑규칙이 그 역을 갖지 않는다면 뷰 갱신문제를 해결해야만 한다. 다음은 CODASYL 스키마로부터 관계형 스키마를 생성하는데 쓰이는 매핑규칙을 단순화한 한 예이다.

- 각 레코드 타입은 같은 이름의 테이블로 매핑된다.
- 레코드 타입 A의 각 항목은 테이블 A의 항목으로 매핑된다.
- 레코드 타입 A의 각 레코드 식별자는 테이블 A의 키로 매핑된다.
- 각 셋 (레코드 타입 A와 레코드 타입 B간의 일대다 관계성)은 테이블 B의 열 (외래키라 불림)로 매핑된다. 여기서 테이블 B는 테이블 A의 키값을 가지고 있다.

CDM과 같은 시맨틱 데이터 모델의 사용은 관계형과 같은 재래식 모델에서는 명시하기 곤란한 부가적 시맨틱의 표현을 쉽게해 줄 수 있다. 예를 들어 관계형 모델에서는 표현하기 곤란한 두 엔티티 타입간 inheritance를 허용하는 일반화 관계성을 시맨틱 데이터모델을 사용하는 구성스키마에서는 명백하게 표현할 수 있다. 다른 예로서 관계형 모델로 표현된 지역스키마에서 릴레이션 R1과 R2간의 외래키는 대응되는EER 모델의 구성스키마에서 R1과 R2를 나타내는 두 엔티티간의 관계성으로서 명백히 표현될 수 있다. 그러나 지역스키마로부터 구성스키마로의 변환동안에 제공된 부가적시맨틱 정보에 대하여는 주의가 필요하다. 왜냐하면 연합에 참여하는 DBMS는 자치적이고 또한 그 DBMS가 관리하는 데이터베이스는 변환과정의 결과로 바뀌는 것이 아니기때문이다. 이는 즉, 변환은 다음과 같은 의미에서 역방향으로도 가능해야 함을 의미한다. 즉, (1) CDM의 구성스키마는 지역스키마에 의해 표현된 같은 데이터베이스를 표현해야 한다. (2) 구성스키마상의 한 명령을 대응되는 지역스키마상의 명령으로 변환하는 것이 가능해야 한다. 이것이 바로 내용보존 (content preserving)이라고 하는 스키마변환의 중요 성질이다.

스키마통합 (Schema Integration)

스키마 통합이란 이미 존재하는 또는 제안된 데이터베이스들을 하나의 통일된 전역 스키마로 통합하는 행위로서 이 전역스키마는 분산 데이터베이스 환경에 참여한 모든 데이터베이스들의 가상 뷰이다. 스키마를 통합할때 수행되는 작업에는 (1) 서로다른 원시스키마에 있는 객체들이 어떻게 연관되어 있는 가를 알아내고 결정하기 위한 스키마와 객체의 분석, (2) 원시스키마에 대한 매핑과 함께 통합된 객체를 생성하기위한 스키마와 객체의 합병 등의 두가지 작업이 있다. 이러한 작업을 지원하기 위해사용되는 테크닉은 스키마 분석과 통합의 목적, 스키마의 데이터 모델, 정보의 완전성, 통합기의 능력 등에 따라 좌우된다. 일반적으로 분석 작업에 대한 기술이 합병 작업에 대한 기술보다 더 어려운 것으로 간주된다. 또한 때로는 하나의 기술만으로는 충분하지 않고 여러 기술이 함께 사용되는 경우도 있다.

스키마통합시 데이터 모델 및 DBMS의 다양성 때문에 발생하는 문제점들을 해결하기 위해서 공통 데이터 모델을 이용하여 원시스키마인 각 지역스키마들을 표현한다. 이 때 쓰이는 데이터 모델로서는 풍부한 시맨틱을 포함하는 시맨틱 데이터 모델이많이 쓰인다. 이러한 시맨틱 데이터 모델에는 Functional Data Model 및 DAPLEX, Entity Relationship(ER)모델, Extended ER (EER)모델 등이 있다. 스키마통합 절차는 보통 다음과 같은 4단계로 이루어진다.

단계 1 : 프레통합

여러 모델을 통합하는 과정은 각각의 원시스키마로부터 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 이미한다.

단계 2 : 스키마 분석

스키마분석 단계에서는 같은 현실 세계를 나타내는 서로 다른 스키마 객체들이 합병될 수 있도록 원시스키마들이 분석되어진다. 이 단계에서 원시스키마 객체들간의 충돌이 분석되고 해결되어야 한다.

충돌 분석은 두 스키마간의 같은 대상을 나타냄에 있어서의 모든 차이를 알아내는 것으로 이는 크게 나누어 스키마내의 개념의 이름을 비교하여 통일시키는 이름 충돌 분석과, 스키마내의

개념의 표현을 비교하여 통일시키는 구조 충돌 분석이 있다.

(a) 이름 충돌 분석

이름 충돌을 야기하는 두가지 근원은 동음이의어(homonym)와 이음동의어(synonym)이다. 이음동의어는 응용 도메인의 한 객체가 두 스키마에서 서로 다른 이름으로 표현되었을때 발생하고 동음이의어는 응용 도메인의 서로 다른 객체가 두 스키마에서 같은 이름으로 표현되었을 때 발생한다. 이음동의어와 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 사로 다른 성질과 제약조건들을 가질때 존재한다. 유사성은 동의어를, 불일치는 동음이의어를 야기한다. 유사개념 및 불일치 개념을 발견해낸 뒤에는 이름의 재부여 같은 스키마를 수정하는 과정이 뒤따르게 된다.

(b) 구조 충돌 분석

구조 충돌 분석시에는 입력스키마에서 같은 이름을 가진 개념들이 합병될 수 있는가를 알아보기 위해 비교된다. 구조 분석은 다음과 같은 범주로 나뉜다.

- 동일 개념(identical concepts): 아주 똑같은 표현구조와 성질을 가짐
- 동등한 개념(compatible concepts) : 다른 표현구조와 성질을 가지나 모순되지 않음
- 동등하지 않은 개념(incompatible concepts) : 모순되는 성질을 가짐

이 중 동등한 개념의 충돌은 두 표현중 하나를 변화시킴으로서 쉽게 해결 가능하다.

그러나 동등치 않은 개념의 충돌은 두 스키마를 충분히 지원하는 공통 표현법에 의해 해결해야 한다.

단계 3 : 합병

이 단계에서는 충돌이 해결된 원시스키마들이 합병된다. 이때 여러 스키마를 한꺼번에 합병하는 것도 가능하나 한번에 두 스키마씩만 통합한다.

단계 4 : 재구성 및 문서화

합병후에라도 부수적인 시맨틱을 반영하기 위해서 합병된 스키마를 재구성할수도 있다. 합병결과를 분석하는 능력은 매우 중요하다. 문서화는 사전 정보와 스키마에 대한 주석을 달고 보고서를 작성하는 작업을 말한다.

5. 요약 및 결론

본 고에서는 데이터베이스 설계를 위한 기초이론과 ER 모델을 이용한 관계형데이터베이스 설계 및 신기술 분야에 있어서의 데이터베이스 설계를 다루었다.

데이터베이스 설계는 데이터베이스 연구 영역중 가장 기본이 되는 분야로 효과적인 데이터베이스 시스템 운용을 위하여서는 좋은 설계가 필수적이다. 왜냐하면 아무리 많은 수고와 노력을 들여 구축한 데이터베이스 시스템이라 할지라도 데이터베이스에 오류가 있거나 문제가 있다면 더 이상 쓰이지 않게 되기 때문이다. 좋은 데이터베이스 설계를 위해서는 데이터베이스 설계 절차를 따라 설계하는 것이 중요하다. 최근 들어서는 분산 데이터베이스 설계나 이질형 데이터베이스 통합을 위한 설계에도 많은 관심과 연구가 이루어지고 있다. 이러한 시스템을 위한 데이터베이스 설계는 그 시스템 특성에 맞는 새로운 설계 기법들과 더불어 계속 연구되어야 할 것이다.