데이터베이스 설계

나민영 / University of Folraida 전산학박사,
육군사관학교 전산학교수

전문가 리포트

차 례

2. 관계형 데이터베이스 설계

2.1 ER 모델을 이용한 개념적 설계

개념적 설계는 사용자 요구사항의 복잡한 분석결과이다. 따라서 스키마는 보통 반복적인 과정을 거쳐 산출된다. 이 절에서는 먼저 개념적 설계시 사용되는 설계도구인 ER모델을 간단히 살펴본 후 이를 이용하여 개념적 설계를 실시하는 과정을 예를 들어 설명한다.

ER 모델

ER(Entity-Relationship) 모델은 1976년 Peter Chen에 의해 소개된 이래 개념적 설계에 가장 많이 사용되는 모델로서 더욱 더 일반적으로 쓰이고 있다. ER모델은 최초에는 엔티티, 관계성, 그리고 애트리뷰트와 같은 기본적인 개념들로 구성되었으나 나중에 일반화 계층구조와 같은 복잡한 개념들이 첨가되어 확장된 ER 모델로 발전하였다. 이제 ER 모델의 기본 요소, 제약조건, 기타요소 및 확장요소 등을 간단히 살펴본다.

● 엔티티(Entity)

엔티티란 현실 세계의 객체를 나타낸다. 엔티티는 실제적으로 존재하는 것일수도 있고 개념적인 것일 수도 있다. 예를 들면 사람, 남자, 여자, 회사, 도시, 학과목 등과 같은 것들이다. 엔티티는 ER 다이아그램에서 사각형으로 표현된다.

● 애트리뷰트(Attribute)

애트리뷰트는 엔티티 또는 관계성의 성질을 나타낸다. 애트리뷰트는 ER 다이아그램에서 작은 원으로 표시된다. 어떤 애트리뷰트들은 독립적인 여러 작은 부분들로 쪼개질 수 있다. 예를 들면 주소 애트리뷰트 같은 것들이다. 주소 애트리뷰트는 적어도 시, 구 동으로 쪼개질 수 있기 때문이다. 이와 같이 여러개의 기본 애트리뷰트들로 구성된 애트리뷰트를 복합 애트리뷰트(composite attribute)라 한다. 또한 어떤 애트리뷰트는 같은 엔티티에 대해 여러개의 값을 가질 수 있다(예 : 자동차 색상). 이와 같은 애트리뷰트들은 다중값(multivalued)애트리뷰트라 한다. ER 다이아그램에서 복합 애트리뷰트는 타원으로, 다중값 애트리뷰트는 이중원으로 표시된다.

● 관계성(Relationship)

관계성은 둘 또는 그 이상 엔티티간의 관계를 나타낸다. 관계성은 ER 다이아그램에서 마름모꼴로 표현된다. 여기서 두 엔티티 간의 관계를 이진관계성(binary relationship)이라 하고, 그 이상 엔티티 간의 관계를 N-ary 관계성이라 하는데 데이터베이스 설계에서는 통상 이진관계성을 사용한다. 관계성 역시 엔티티와 마찬가지로 애트리뷰트를 가질 수 있다.

● 카디날리티 제약조건(Cardinality Constraint)

카디날리티 제약조건이란 매핑 제약조건이라고도 하는데 이진 관계성에 연관된 두 엔티티간에 있어서 엔티티의 원소가 참여할 수 있는 수에 대한 제약조건을 말한다. 카디날리티 제약조건에서는 일대일(1:1), 일대다(1:N), 다대다(M:N) 등의 형태가 있다.

● 키(Key)

엔티티에 있어 중요한 제약조건중 하나는 키 제약조건이다. 엔티티 E의 키란 E의 모든 인스턴스들을 unique하게 결정하는 특성을 가진 애트리뷰트나 애트리뷰트의 셑을 말한다. 이러한 특성을 가진 애트리뷰트나 애트리뷰트의 셑이 다수인 경우 이들을 모두 가리킬때 후보키(Candidate Key)라는 용어를 사용한다. 후보키중에서 특별히 식별용으로 선정된 키를 주키(Primary Key)라 하고, 주키로 선정되고 남은 키를 부키(Alternate Key)라 한다. 키는 하나의 애트리뷰트로만 이루어지는 것이 아니라 여러 애트리뷰트의 복합으로도 구성될 수 있다. 이렇게 애트리뷰트가 복합되어 이루어진 키를 복합키라 한다. 키 애트리뷰트는 다이아그램에서 까만 원으로 표시된다.

● 일반화 계층(Generalization Hierarchy)

ER 모델의 일반화 계층은 추상화 개념으로서의 일반화를 나타낸다. 일반화란 여러 클래스 간의 공통적인 특성을 파악하는 과정을 말한다. 예를 들어 생물은 동물의 일반화이다. 왜냐 하면 모든 동물은 생물이기 때문이다. 즉 IS-A 관계를 만족하기 때문이다. 마찬가지로 클래스 운송기구는 클래스 승용차와 클래스 트럭의 일반화이다. ER 다이아그램에서 화살표는 일반화된 엔티티를 가리킨다. 일반화의 가장 유용한 기본적인 특성은 상속(inheritance) 특성으로서 이는 루트엔티티(generic 엔티티)에 있는 모든 애트리뷰트가 자동적으로 서브셑 엔티티로 상속되어짐을 의미한다.

일반화에 반대되는 개념은 Specialization이다. Specialization이란 한 엔티티의 서브클래스의 셑을 정의하는 과정을 말한다. Generalization/Specialization 관계는 수퍼클래스/서브클래스 관계라고도 한다.

개념스키마 설계

이제 ER모델을 이용하여 개념스키마를 설계해 보자. 개념스키마 설계는 먼저 스키마의 골격을 잡는 일부터 시작된다. 이때 작성된 스키마를 골격스키마(Skeleton schema)라 하는데 골격스키마에 나타나는 개념은 가장 명백한 개념이다. 먼저 요구사항 분석시 결정된 주요 개념이 골격스키마의 ER 모델의 엔티티가 되도록 한다. 엔티니가 선택되면 그 엔티티를 중심으로 관계성을 부여한다. 엔티티와 관계성이 추출되면 요구사항에서 명시된 모든 기능들을 표현하기 위해 스키마를 다듬고 확장시킨다. 이를 위해 골격스키마에 있는 개념들이 더 다듬어질 수 있는지를 분석해서 스키마가 완전히 정제되면 나머지 애트리뷰트, 키, 매핑 등을 첨가해서 개념스키마를 완성한다.

개념스키마를 작성하는 과정의 이해를 돕기 위해 예를 들어 설명한다. 다음의 예는 여러 부서에서 여러 프로젝트를 수행하는 회사업무에 대한 데이터베이스를 설계하는 것이다. 먼저 수집 분석된 요구사항을 정리한 결과는 다음과 같다.

1. 회사는 부서로 조직되어있다. 각 부서는 부서명, 부서번호, 부서장에 관한 정보를 유지한다. 또한 부서장이 취임한 날짜도 유지한다. 한 부서는 여러 곳에 위치할 수도 있다.
2. 부서는 여러 프로젝트를 통제한다. 각 프로젝트는 프로젝트명, 프로젝트 번호, 프로젝트가 수행되는 장소(단일장소)에 과한 정보가 있어야 한다.
3. 한 회사원에 대해서는 주민등록번호, 주소, 봉급, 성별, 생년월일 등을 유지한다. 회사원은 각각 한 부서에 소속되나 여러 프로젝트에서 일할 수 있다. 이때 이 여러 프로젝트

는 같은 부서에 의해서 통제되지 않을 수도 있다. 따라서 각 회사원이 각 프로젝트에서 당 근무하는 주당 근무시간을 유지해야 한다.

4. 회사원은 그 역할에 따라 기술자와 사무원으로 구분된다. 기술자에 대해서는 기술등급을, 사무원에 대해서는 타이핑 속도에 관한 정보를 각각 유지한다.

분석된 요구사항으로부터 엔티티와 애트리뷰트를 추출하면 다음과 같다. {}는 다중값 애트리뷰트를 나타낸다.

부서 (부서명, 부서번호, {위치}, 부서장, 취임일)
프로젝트 (이름, 번호, 위치, 통제부서)
회사원 (이름, 주민등록번호, 성별, 주소, 봉급, 생년월일, 부서, {작업(프로젝트, 근무시간)})

또한 회사원에는 일반화 계층이 존재한다. 즉 회사원 엔티티는 그 서브클래스로 기술자 엔티티와 사무원 엔티티를 가진다. 추출된 엔티티 간에는 관계성이 존재함을 쉽게 알 수 있다. 엔티티 간의 관계성 추출을 위해 엔티티들을 분석해 보면 다음과 같다.

(1) 회사원과 부서

회사원과 부서 사이엔ㄴ 두개의 관계성이 존재한다. 첫번째는 한 부서에 여러명이 근무하게 되므로 다대일(N:1)의 '근무' 관계성이다. 두번째는 부서마다 부서장이 부서를 관리하므로 일대일(1:1)의 '관리' 관계성이다.

(2) 부서와 프로젝트

부서와 프로젝트와의 사이에는 한 부서가 여러 프로젝트를 통제할 수 있으므로 일대다(1:N)의 '통제' 관계성이 있다.

(3) 회사원과 프로젝트

회사원과 프로젝트 사이에는 한 회사원이 여러 프로젝트에서 일할 수 있고 한 프로젝트는 여러명의 회사원이 함께 일할 수 있으므로 다대다(N:N)의 '작업' 관계성이 있다.

지금까지 추출된 엔티티, 애트리뷰트, 관계성을 ER 다이아그램을 이용하여 표현하면 그림 6과 같다. 이때 회사원내의 '작업'이란 복합다중값 애트리뷰트가 다대다의 관계성으로 표현됨을 관심있게 살펴보아야 한다.

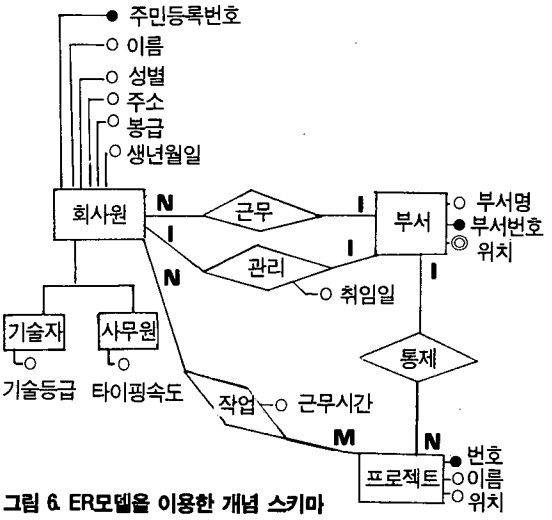

그림 6. ER모델을 이용한 개념 스키마

2.2 관계형 스키마 생성

이 절에서는 ER모델을 이용하여 작성된 개념 스키마로부터 여러개의 릴레이션 정의로 구성되는 관계형 스키마를 얻는 방법을 다룬다. 이 단계로부터 산출되는 산출되는 결과는 여러개의 릴레이션의 정의로 구성되는 관계스키마이다. 따라서 먼저 관계형 모델을 간단히 살펴본 후 ER 모델로부터 관계형스키마를 얻는 방법을 설명한다.

관계형 모델

관계형 모델은 데이터베이스의 데이터를 릴레이션으로 표현한다. 릴레이션이란 일종의 테이블

로서 각 행은 관련된 데이터 값들을 나타내고 각 열은 각 행에 있는 해당되는 값들이 해석되어지는 근거를 제공한다. 예를 들어, 그림 7의 회사원 릴레이션에서 첫번째 회사원의 데이터 값을 보면 <1001, 580423-1027543, 36, 총무과>이다. 이 행의 의미는 사번이 1001, 주민등록번호는 580423-1027543, 나이는 36, 그리고 근무부서는 총무과라는 것이다. 이러한 열과 행을 데이터베이스 용어로 나타내면 각 열은 애트리뷰트(attribute), 행은 투플(tuple)이라 하며, 투플의 수를 카디날리티라 한다. 그리고 애트리뷰트가 가질 수 있는 값의 셑을 도메인(domain)이라 한다. 예를 들어 이 회사에 부서가 총무과, 인사과, 자재과, 영어과 뿐이라면 부서 애드리뷰트의 도메인은 <총무과, 인사과, 자재과, 영업과>가 된다.

사번	주민등록번호	나이	근무부서
1001	580423-1027543	37	총무과
1002	600110-1027555	35	자재과
1003	651001-1025222	30	영업과
.			
.			

그림7. 회사원 릴레이션

릴레이션은 투플의 셑으로 정의될 수 있다. 이것은 즉, 같은 두개의 투플은 존재하지 않는다는 말이다. 릴레이션 R의 모든 투플들은 unique하게 식별할 수 있는 특성을 가진 애트리뷰트나 애트리뷰트의 집합을 키(key)라 한다. 키에 대한 기본 용어 및 정의는 ER모델에서 설명한 것과 마찬가지이고 추가적으로 외래키가 있다. 외래키(외부키, foreign key)란 지금의 릴레이션에서느 키로 쓰이지 않지만 다른 릴레이션에서는 키로 쓰이고 있는 애트리뷰트를 말한다. 왜래키는 조인시 자주 사용되는 중요한 개념이다. 이 관계형 모델을 위한 제약조건과 데이터베이스 언어를 살펴보자.

● 제약조건

제약조건이란 데이터베이스 스키마가 제대로 데이터베이스 인스턴스들을 유지할 수 있도록 하기 위한 일종의 강제적 규정이다. 관계형 데이터베이스 제약조건에는 다음과 같은 것들이 있다.

- 키 제약조건 : 키의 값은 릴레이션 내의 어느 투플에서도 유일해야 한다.
- 엔티티 무결성 제약조건 : 주키 값은 널(null) 일수 없다는 제약조건이다. 주키 값이 널이면 식별할 수 없는 투플이 발생하게 되기 때문이다.
- 참조 무결성 제약조건 : 참조 무결성 제약조건은 두 릴레이션간 투플사이에 일간성을 유지하기 위한 제약조건으로서 한 릴레이션 A내에 있는 투플이 다른 릴레이션 B에 있는 투플을 참조할때는 참조되는 이 투플이 반드시 그 릴레이션 B내에 존재하여야 한다는 제약조건이다.

● 관계형 데이터베이스 언어

데이터베이스 언어에는 여러가지가 있으나 최근 관계형 DBMS에서 가장 많이 쓰이는 언어는 SQL 타입의 언어이다. SQL은 Structured Query Language의 약자로서 IBM 연구소에서 System R의 인터페이스로 개발된 이래로 지금은 많은 버젼들이 여러 DBMS 개발회사들에 의해 개발되어 사용되고 있다. SQL은 지금 IBM의 DB/2와 SQL/DS 라는 상용제품의 언어로서 사용되고 있다. SQL에서는 릴레이션, 투플, 애트리뷰트란 용어 대신 테이블, 행, 열이란 용어를 각각 사용하나 그 의미는 데이터베이스 전문용어와 같은 의미로 사용되고 있다. SQL과 같은 데이터베이스 언어는 크게 DDL(Data Definition Language)과 DML(Data Manipulation Language)로 나뉜다. DDL은 릴레이션의 생성 및 소멸에 관련된 명령이고 DML은 데이터조작에 관련된 명령이므로 여기서는 DML, 그중에서도 특히 질의어를 다룬다. 질의어(Query Language)란 stand alone 형태의 대화식 방법으로 사용되는 고급 DML을 가리킨다.

데이터베이스로부터 정보를 검색하는 기본적인

문장은 SELECT문이다. SELECT문은 다음과 같이 SELECT, FROM, WHERE의 세절로 구성된다.

SELECT <애트리뷰트 리스트>
FROM <테이블 리스트>
WHERE <조건>

여기서 <애트리뷰트 리스트>는 질의어에 의해 검색되는 값과 관련된 애트리뷰트들이고, <테이블 리스트>는 질의어를 처리하기 위해 요구되는 릴레이션의 리스트들이고, <조건>은 질의어에 의해 검색되어지는 투플들을 식별해내기 위한 조건적 표현이다.

ER모델로부터 관계형스키마로의 변환

ER 스키마를 관계형 스키마로 변환하는 작업은 먼저 복합 애트리뷰트와 다중값 애트리뷰트를 제거하고 난후 각 엔티티를 릴레이션으로 변환하고, 각 관계성을 그 카디날리티 제약조건에 따라 처리해 줌으로써 이루어진다.

● 복합 애트리뷰트 및 다중값 애트리뷰트의 제거

관계형 모델은 단순하고 단일값을 갖는 것을 그 기초로 한다. 따라서 복합 애트리뷰트나 다중값 애트리뷰트는 단순 애트리뷰트나 단일값 애트리뷰트의 형태로 바뀌어야 한다. 복합된 애트리뷰트에 대해서는 다음 두가지 방법으로 해결할 수 있다.

(1) 복합 애트리뷰트를 구성하는 모든 구성원들을 각각 애트리뷰트로 하거나
(2) 복합 애트리뷰트의 개별 구성 요소들을 제거해 버리고 복합 애트리뷰트를 단순 애트리뷰트로 간주한다.

다중값 애트리뷰트는 새로운 엔티티를 필요로 한다. 각 다중값 애트리뷰트는 그 값이 단순 애트리뷰트로 표현될 수 있는 엔티티가 있어야 한다. 새로운 엔티티는 다중값 애트리뷰트와 원래 엔티티의 키로 구성되며 이때 새로운 엔티티는 다중값 애트리뷰트와 원래 엔티티의 키로 구성되며 이때 새로운 엔티티의 키는 모든 애트리뷰트이 셑으로 이루어진다.

● 엔티티 변환

이 과정은 비교적 단순하다. 스키마의 각 엔티티는 릴레이션으로 변환된다. 따라서 엔티티의 애트리뷰트와 주키는 릴레이션의 애트리뷰트와 주키로 된다.

● 일대일 관계성의 변환

먼저 일반적인 방법인 이진 관계성을 생각해 보자. 원칙적으로 일대일 관계성에 참여한 두 엔티티 E1, E2는 별도의 릴레이션을 생성해내고 관계성은 그 중 한 엔티티에 흡수된다. 그러나 경우 따라 하나의 릴레이션으로 압축될 수도 있다. 따라서 다음과 같은 경우로 나뉜다.

(1) 하나의 릴레이션으로 압축 : 이 경우는 두 엔티티가 같은 주키를 갖는 경우이다. 이런 경우에는 이 두 엔티티에 해당되는 두 릴레이션이 하나의 일레이션으로 압축되고 모든 애트리뷰트들이 통합되어진다.
(2) 관계성을 흡수한 두개의 릴레이션 생성 : 두 엔티티가 서로 다른 주키를 가지며 그 관계성이 일대일인 경우이다. 이때는 두 엔티티에 해당하는 두 릴레이션이 생성되고 어느쪽이든지 한 릴레이션의 주키를 다른 릴레이션에 포함시켜 외부키로 유지하면 된다.

● 일대다 관계성의 변환

이제는 두 엔티티 E1과 E2의 관계성이 일대다(1:N)인 경우를 생각해 보자. 이 관계성에서 E1은 1편으로, E2는 N편으로 참여한다고 하자. 그러면 각 엔티티가 릴레이션으로 변환될때 1편 엔티티인 E1의 주키는 N편 엔티티인 E2의 한 애트리뷰트(또는 애트리뷰트의 셑)으로 첨가되어야 한다.

● 다대다 관계성의 변환

다대다 관계성을 변환하기 위해서는 일대일, 일대다 관계성의 경우와는 달리 별도의 릴레이션을 만들어 주어야 하다. 다대다 관계성에 참여하는 두 엔티티를 E1, E2라 하면 E1과 E2에 해당하는 릴레이션을 각각 만든후 관계성을 나타내는 별도의 릴레이션 E3를 만들어 E1의 주키와 E2의 주키로 구성되는 복합키를 형성하게 한다. 이때 관계성에 딸린 애트리뷰트는 관계성을 나타내는 별도 릴레이션에 애드리뷰트로 포함됨을 유의해야 한다.

● 순환 관계성의 매핑

엔티티 E로부터 자기 자신에게 관계되는 순환 관계성은 두 애트리뷰트를 포함하는 새로운 릴레이션을 만들거나 엔티티 E에 새로운 애트리뷰트를 추가함으로 매핑을 이룰 수 있다. 새로운 릴레이션을 만드는 경우에 있어서 이 두 애트리뷰트는 모두 E의 주키에 해당되는 것으로 그 이름은 E의 역할이름에 상응한다. 이때 관계성 형태가 일대일, 일대다, 다대다 중 어느것인가에 따라 이들중 하나 또는 둘다가 새로운 릴레이션의 주키로 선정된다.

● 수퍼클래스/서브클래스 매핑

수퍼클래스/서브클래스 관계성을 매핑하는 방법에는 여러가지가 있다. 여기서는 이중 많이 쓰이는 두가지 방법만 소개한다. 먼저 수퍼클래스/서브클래스 관계성에 참여하는 서브클래스들을 Si, 1≤i≤m, 라하고 수퍼클래스를 C라 한다.

옵션1 : 수퍼클래스 C에 대해 하나의 릴레이션을 생성하고 서브클래스들에 대해서도 각각 릴레이션을 하나씩 생성한다. 이때 C의 주키가 서브클래스에 해당하는 릴레이션 각각에 첨가되어 그 릴레이션에서 주키로 사용된다. 이렇게 함으로서 수퍼클래스로부터의 애트리뷰트 상속 개념이 지원 가능하게 된다.

옵션2 : 수퍼클래스에 대해서는 별도의 릴레이션을 생성하지 않고 서브클래스들에 대해서만 각각 릴레이션을 생선한다. 이때 생성된 릴레이션의 애트리뷰트는 서브클래스 자신의 애트리뷰트와 수퍼클래스의 애트리뷰트들로 구성되며 수퍼클래스에서 주키로 사용되었던 애트리뷰트가 새로 생성된 릴레이션에서도 계속 주키로 사용된다.

● 변환 예

지금까지 설명된 변화기법을 이용하여 앞절에서 설계된 ER스키마를 관계형 스키마로 변환하면 다음 그림 8과 같은 결과를 얻는다. 이때 변환된 결과에서 밑줄친 애트리뷰트는 키 애트리뷰트임을 의미하고 음영으로 표시된 애트리뷰트는 외래키임을 의미한다. 여기서 일대다의 '근무' 관계성은 부서번호를 N편 릴레이션인 회사원 릴레이션에 하나의 애트리뷰트외래키로서 첨가하므로 그 시맨틱이 유지되고 일대일 관계성인 '관리'는 부서 릴레이션에 '부서장주민번호'라는 애트리뷰트로서 표현되어 있음을 알 수 있다. 또한 다대다 관계성인 '작업'은 별도의 릴레이션인 작업 릴레이션으로 변환되어지고 이때의 키는 회사원의 주민등록번호와 프로젝트의 프로젝트번호로 구성되는 복합키이며 관계성에 딸린 애트리뷰트가 이 작업 릴레이션의 애트리뷰트로 포함됨을 유의해야 한다. ER 스키마로부터 관계형 스키마로 매핑이 끝나면 결정된 DBMS의 DDL을 이용하여 데이터베이스를 생성하게 된다.

회사원(주민등록번호, 이름, 성별, 주소 봉급, 생년월일, 부서번호)
기술자(주민등록번호, 기술등급)
사무원(주민등록번호, 타이핑속도)
부서(부서번호, 부서명, 부서장주민번호, 취임일)
부서장소(부서번호, 부서위치)
프로젝트(프로젝트번호, 이름, 위치, 부서번호)

작업(주민등록번호,프로젝트번호, 근무시간)

그림& 생성된 관계형 스키마

정규화(Normalization)

잘못된 관계형 스키마 설계는 각종 이상현상(Anomaly)을 유발하여 애써 구축한 데이터베이스를 망치게 하는 요인이 되므로 올바른 설계를 하는 것은 매우 중요하다. 정규화란 함수족 종속성(Functional Dependency)등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 애트리뷰티 셑으로 쪼개어 바람직한 스키마, 즉 정규형(Normal Form)으로 만들어 가는 과정이다. 정규형에는 제1정규형, 제2정규형, 제3정규형, BCNF형, 제4정규형, 그리고 제5정규형 등 여러 종류가 있어 이 순서대로 더욱 완전한 정규화를 지원해준다. 그러나 관계형 스키마는 이중 제3정규형 또는 BCNF형 까지만 만족하면 되는 것으로 알려져 있다. 일반적으로 ER 모델링을 이용하여 현실세계의 시맨틱을 제대로 표현하였다면 대체로 제3정규형을 만족하게 되므로 더 이상의 특별한 정규화 작업이 필요하게 된다. 정규화에 관한 자세한 이론은 지면관계상 생략하고 여기서는 단지 정규화의 기초로서 잘못된 스키마로 인한 각종 이상현상과 분할원칙만을 살펴본다.

● 잘못된 스키마로 인한 이상 현상

다음과 같은 관계성 스키마가 있다고 하자. 이 스키마는 잘못된 스키마로서 정보의 중복이 초래되어 다음과 같은 이상현상들이 발생하게 된다.
회사원(사번, 성명, 생년월일, 주소, 부서번호, 부서명, 부서장)

(1) 삽입이상(Insertion Anomaly)
- 새로운 회사원 투플을 회사원 테이블에 삽입하려면 부서에 대한 애트리뷰트 값도 알아야만 한다. 그렇지 않으면 null값이 들어가게 된다.
- 회사원이 아직 없는 새로운 부서를 삽입하기가 아주 곤란하다. 오직 한가지 방법은 null값을 넣는것인데 이는 주키가 null이 되는 문제가 발생한다.

(2) 삭제 이상(Deletion Anolmaly)
- 그 부서에서 일하는 마지막 회사원을 삭제했을 때에는 그 부서에 대한 정보자체도 없어져 버린다.

(3) 갱신 이상(Update Anomaly)
- 한 부서의 애트리뷰트중 하나의 값을 변경시키려 할때 그 부서에서 일하는 모든 회사원에 대해서 수정해야만 한다.

따라서 이러한 이상현상들을 해소하려면 스키마를 둘로 쪼개어 다음과 같은 두개의 스키마로 구성해야 한다.

회사원(사번, 성명, 생년월일, 주소, 부서번호)
부서(부서번호, 부서명, 부서장)

● 스키마 분할 원칙

함수적 종속성을 고려하여 정규화를 실시할때 스키마를 쪼개는 경우가 자주 발생하는데 스키마 분할시에는 다음과 같은 원칙을 준수해야 한다.

(1) Lossless-Join 분할

하나의 테이블을 둘로 쪼개었을때 이들을 조인하면 정보의 손실이 없이 원래의 테이블이 생성되어야 한다. 여기서 정보의 손실이란 투플이 줄어드는 경우뿐만 아니라 가짜 투플로 인한 투플의 증가까지도 포함한다.

(2) Dependency Preservation

테이블 스키마 내에 내재된 함수적 종속성은 테이블이 쪼개지기 전이나 후에나 마찬가지로 똑같이 유지되어야 한다.