데이터베이스

1. 데이터베이스 시스템
   1) 정보화 사회에 정보를 수집, 관리하고 분석하는데 데이터베이스 기술이 주요하게 활용되고 있음.
   2) 용어정의
      (1) 데이터베이스
      (2) 데이터베이스 관리 시스템
      (3) 사용자 및 관련 응용 프로그램
   3) 데이터베이스 정의 및 개요
      (1) 데이터베이스는 조직체의 응용 시스템들이 공유해서 사용하는 운영 데이터들을 구조적으로 통합하여 저장한 (의 모임이다.(이건 정의 개요 두 개다 들어가는 항목임)
      (2) 시스템 카탈로그와 저장된 데이터베이스로 구분할 수 있음.
      (3) 시스템 카탈로그는 저장된 데이터베이스의 스키마 정보(메타 정보)를 유지
   4) 데이터베이스 특징
      (1) 동시 공유 - 데이터베이스는 데이터의 대규모 저장소로서 여러 부서에 속하는 여러 사용자에 의해 동시에 사용됨
      (2) 중복 최소화 - 모든 데이터가 중복을 최소화 하면서 통합됨
      (3) 자기 기술 - 한 조직체의 운영 데이터(일반데이터인 듯?)뿐만 아니라 그 데이터에 관한 설명(메타데이터)까지 포함
      (4) 데이터 독립성 - 프로그램과 데이터 간의 독립성이 제공됨 (논리스키마가 바뀌어도 외부스키마가 바뀌지 않으며, 물리스키마가 바뀌어도 개념스키마가 바뀌지 않는다. 즉 ER-WIN이 바뀌어도 프로그램에는 아무 영향이 없고, SQL이 바뀌어도 ER-WIN에는 아무 영향이 없다. 요소리인 듯.) (개요부분임)
      (5) 효율적인 처리 - 효율적으로 접근이 가능하고 질의를 할 수 있음. (이거 개요부분)
   5) 데이터베이스 관리 시스템 -DBMS
      (1) 데이터베이스와 사용자간의 인터페이스 기능
      (2) 데이터베이스를 정의하고, 질의 처리를 지원하고, 접근관리기능 등의 작업을 수행하는 소프트웨어.(요까지가 특징인 것 같음)
      (3) 사용자가 새로운 데이터베이스를 생성하고, 데이터베이스의 구조를 명세할 수 있게 하고, 사용자가 데이터를 효율적으로 질의하고 수정할 수 있도록 하며, 시스템의 고장이나 권한이 없는 사용자로부터 데이터를 안전하게 보호하며, 동시에 여러 사용자가 데이터베이스를 접근하는 것을 제어하는 소프트웨어 패키지.
      (4) 데이터베이스 언어라고 부르는 특별한 프로그래밍 언어를 한 개 이상 제공
      (5) SQL은 여러 DBMS에서 제공되는 사실상의 표준 데이터베이스 언어.
   6) 하드웨어
      (1) 데이터베이스는 디스크와 같은 보조 기억 장치에 저장되며, DBMS에서 원하는 정

보를 찾기 위해서는 디스크의 블록들을 주기억장치로 읽어 들여야 하며, 계산이나 비교 연산들을 수행하기 위해 중앙 처리 장치가 사용됨.

(2) DBMS자체도 주기억 장치에 적재되어 실행되어야 함.

7) 데이터베이스 시스템 요구사항

(1) 데이터 독립성

(2) 효율적인 데이터 접근

(3) 데이터에 대한 동시 접근

(4) 백업과 회복

(5) 쉬운 질의어

(6) 다양한 사용자 인터페이스

(7) 중복을 줄이거나 제어하며 일관성 유지

(8) 데이터 무결성

(9) 데이터 보안

2. 파일 시스템 VS DBMS

1) 파일시스템을 사용한 기존의 데이터 관리

(1) 파일의 기본적 구성요소는 순차적인 레코드들 (레코드란 테이블이랑 비슷한 뜻)

(2) 하나의 레코드는 연관된 필드의 모임

(3) 파일을 접근하는 방식이 응용프로그램 내에 상세하게 표현되므로 데이터에 대한 응용프로그램의 의존도가 높음.

2) DBMS를 사용한 데이터베이스 관리

(1) 여러 사용자와 응용 프로그램들이 데이터베이스를 공유

(2) 사용자의 질의를 빠르게 수행할 수 있는 인덱스 등의 접근 경로를 DBMS가 자동적으로 선택하여 수행

(3) 권한이 없는 사용자로부터 데이터베이스를 보호

(4) 여러 사용자에게 적합한 다양한 인터페이스를 제공

(5) 데이터간의 복잡한 관계를 표현하며, 무결성 제약조건을 DBMS가 자동적으로 유지 (무결성 제약조건 pdf참조하세용)

(6) 시스템이 고장나면 데이터베이스를 고장 전의 일관된 상태로 회복시킴

(7) 프로그램에 영향을 주지 않으면서 데이터베이스 구조를 변경할 수 있음. (프로그램 - 데이터 독립성)

3) DBMS의 장점

(1) 중복성과 불일치가 감소됨. (데이터 중복은 공간낭비, 데이터 상호 불일치(파일시스템에서 일어날 수 있음))

(2) 시스템을 개발하고 유지하는 비용이 감소됨 (추가적인 운영비용 - 운영프로그램들)

(3) 표준화를 시행하기가 용이 (데이터 표준화 NOT DB)

(4) 보안이 향상됨 (정보를 제한하여 권한관리)

(5) 무결성이 향상됨 - real world의 제약조건을 메타DB에 저장 (예로 학점21학점)
(6) 조직체의 요구사항을 식별 가능
(7) 다양한 유형의 고장으로부터 데이터베이스를 회복 가능 (백업은 DBA가 해야함. 은행점검시간 등,
(8) 데이터베이스의 공유와 동시 접근이 가능함.

4) DBMS 선정시 고려사항
(1) 기술적 요인
① DBMS에 사용되고 있는 데이터 모델, DBMS가 지원하는 사용자 인터페이스, 프로그래밍 언어(다양한 언어), 응용 개발 도구, 저장구조, 성능, 접근 방법 등.
(2) 경제적 요인
① 소프트웨어와 하드웨어 구입 비용, 유지 보수 비용, 직원들의 교육 지원 등.

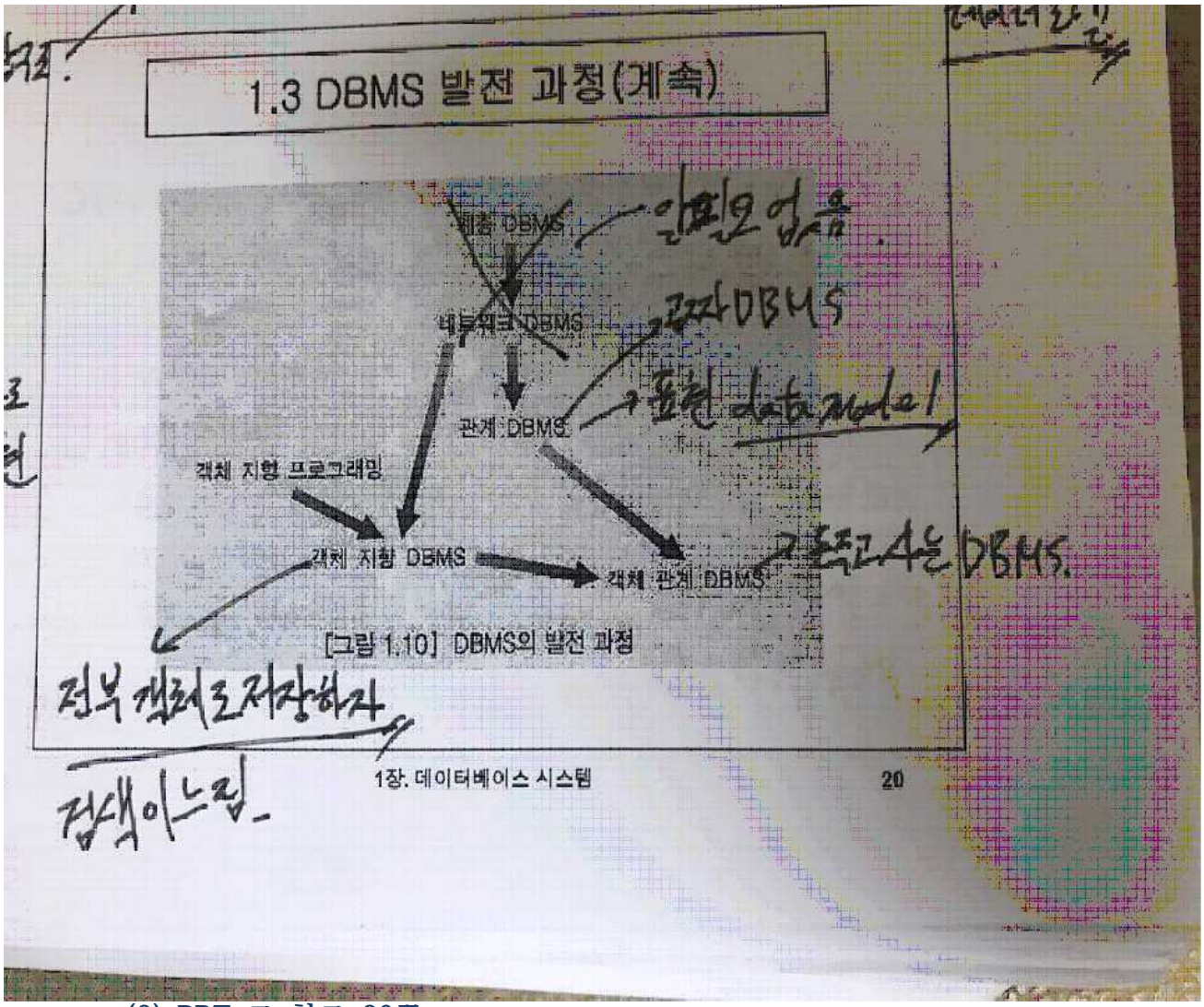
3. DBMS 발전 과정
1) 데이터 모델
(1) 데이터베이스 구조를 기술하는데 사용되는 개념들의 집합인 구조(데이터 타입과 관계), 이 구조 위에서 동작하는 연산자들, 무결성 제약조건들
(2) 사용자에게 내부 저장 방식의 세세한 사항은 숨기면서 데이터에 대한 직관적인 뷰를 제공하는 것과 그 동시에 이들 간의 사상을 제공(데이터 추상화)

[그림 1.9] 관계 데이터 모델에서의 실세계 표현

(3) 위 그림은 실세계를 추상적으로 표현한 테이블. 관계형 데이터 모델.

2) 데이터 모델의 분류
(1) 고수준 또는 개념적 데이터 모델 (현실적)
① 사람이 인식하는 것과 유사하게 데이터베이스의 전체적인 논리적 구조를 명시. 예: 엔티티 관계 데이터 모델과 객체 지향 데이터 모델.
(2) 표현(구현) 데이터 모델
① 최종 사용자가 이해하는 개념이면서 컴퓨터 냉서 데이터가 조작되는 방식과 멀리 떨어져 있지는 않음. 예) 계층 데이터 모델, 네트워크 데이터 모델, 관계 데이터 모델. (구글링 설명 : 논리적 데이터 모델은 개념적 모델링 과정에서 얻은 개념적 구조를 컴퓨터가 이해하고 처리할 수 있는 컴퓨터 세계의 환경에 맞도록 변환하는 과정(아마도 SQL?) 논리적 데이터 모델은 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계를 이용하여 현실 세계를 표현한다. 단순히 데이

터 모델이라고 하면 논리적 데이터 모델을 의미한다. 특정 DBMS는 특정 논리적 데이터 모델 하나만 선정하여 사용한다. 논리적 데이터 모델은 데이터 간의 관계를 어떻게 표현하냐에 따라, 관계 모델, 계층 모델, 네트워크 모델로 구분한다.)



(3) PPT 표 참조 20쪽

(4) 저수준 또는 물리적인 데이터 모델 (하드디스크적)

① 데이터베이스에 데이터가 어떻게 저장되는가를 기술 예) Unifying, ISAM, VSAM 등.

3) 관계 DBMS

(1) E.F. Codd가 IBM연구소에서 관계 ㄷ이터 모델을 제안

(2) 미국 IBM연구소에서 진행된 System R과 캘리포니아 버클리대에서 진행된 Ingres 프로젝트

(3) 장점

① 모델이 간단하여 이해하기 쉬움(수학적 개념으로부터 출발)
② 사용자는 자신이 원하는 것만 명시하고, 데이터가 어디에 있는지, 어떻게 접근해야 하는지는 DBMS가 결정 (예, MS SQL Server, Oracle, Sybase, DB2, MySQL 등)

4) 객체 지향 DBMS
(1) 1980년 등장, 객체 지향 프로그래밍 패러다임을 기반으로 함.
(2) 장점
① 데이터와 프로그램을 그룹화하고, 복잡한 객체들을 이해하기 쉬우며, 유지와 변경이 용이함.
(3) 객체 관계 DBMS
① 1990년 후반 등장, DBMS에 객체 지향 개념을 통합한 객체 관계 데이터 모델이 제안됨.

4. 데이터베이스 시스템 개요
1) 데이터베이스 스키마
(1) 전체적인 데이터베이스 구조를 뜻하며 자주 변경되지 않음.
(2) 데이터베이스의 모든 가능한 상태를 미리 정의
2) 데이터베이스 인스턴스
(1) 특정 시점의 데이터베이스의 내용을 의미하며, 현실세계의 내용을 반영하기 위해 계속 바뀜.

5. DBMS 언어
1) 데이터 정의어 (DDL)
(1) 데이터 정의어로 명시된 문장이 입력되면 DBMS는 사용자가 정의한 스키마에 대한 명세를 시스템 카탈로그 또는 데이터 사전에 저장.(메타DB인 듯)
(2) 기본적인 기능
① 생성 - SQL에서 create table
② 변경 - SQL에서 alter table
③ 삭제 - SQL에서 drop table
④ 인덱스 정의 - SQL에서 create index
2) 데이터 조작어 DML
(1) 데이터 조작어를 사용하여 데이터베이스 인스턴스 내의 원하는 데이터를 검색하고, 수정하고, 삽입하고, 삭제
(2) 절차적 언어와 비절차적 언어-> (기본적으로 선언적인 언어)
(3) 관계 DBMS에서 사용되는 SQL은 대표적인 비절차적 언어
(4) 대부분의 데이터 조작어는 SUM, COUNT, AVG와 같은 내장 함수들을 갖고 있음
(5) 데이터 조작어는 단말기에서 대화식으로 입력되어 수행되거나 C, 코볼등의 고급 프로그래밍 언어로 작성된 프로그램에 내포되어 사용됨.

(6) 기본적인 기능

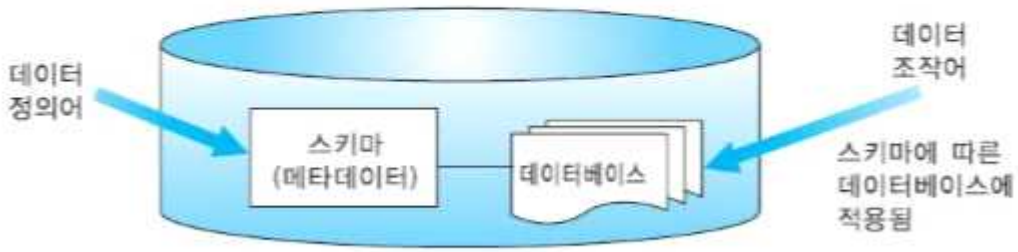
① 데이터 검색 - SQL에서 SELECT

② 데이터 수정 - SQL에서 UPDATE

③ 데이터 삭제 - SQL에서 DELETE

④ 데이터 삽입 - SQL에서 INSERT

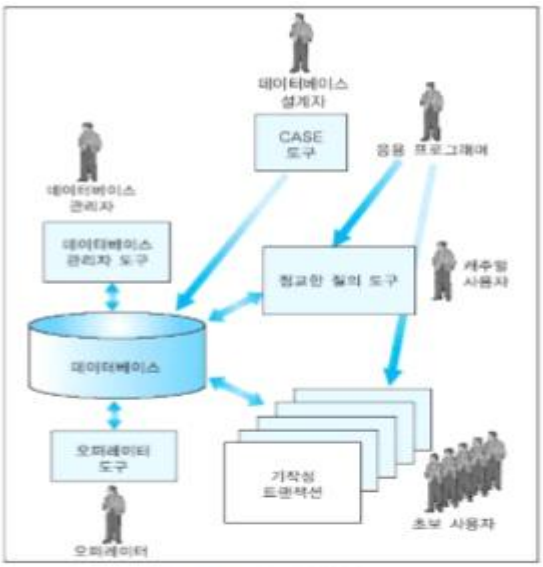
1.4 DBMS 언어(계속)



[그림 1.15] 데이터 정의어와 데이터 조작어

3) 데이터 제어어 DCL

(1) 데이터 제어어를 사용하여 데이터베이스 트랜잭션을 명시하고 권한을 부여하거나 취소.



[그림 1.17] DBMS의 사용자들

6. DBMS 사용자

1) 데이터베이스 설계자

(1) ERWin 등의 CASE 도구들을 이용해서 데이터베이스 설계를 담당
(2) 데이터베이스의 일관성을 유지하기 위해 정규화를 수행

2) 응용 프로그래머 (인터렉티브 유저?)
(1) 데이터베이스 위에서 특정 응용(고객 관리, 인사 관리, 재고 관리 등)이나 인터페이스를 구현하는 사람.
(2) 고급 프로그래밍 언어인 C, 코볼 등으로 응용 프로그램을 개발하면서 데이터베이스를 접근하는 부분은 내표된 데이터 조작어를 사용.
(3) 이들이 작성한 프로그램은 최종사용자들이 반복해서 수행하므로 기작성 트랜잭션이라 부름

3) 데이터베이스 관리자 (DBA : Database Administrator)
(1) 조직의 여러 부분의 상이한 요구를 만족시키기 위해서 일관성 있는 데이터베이스 스키마를 생성하고 유지하는 사람 혹은 팀
(2) 데이터베이스 관리자 역할
① 데이터베이스 스키마 생성 및 변경
② 무결성 제약조건을 명시
③ 사용자의 권한을 허용하거나 취소하고, 사용자의 역할을 관리
④ 저장 구조와 접근 방법(물리적 스키마) 정의
⑤ 백업과 회복
⑥ 표준화 시행

4) 최종 사용자
(1) 질의하거나 갱신하거나 보고서를 생성하기 위해서 데이터베이스를 사용하는 사람
(2) 최종사용자는 2가지 타입으로 구분하는데, 데이터베이스 질의어를 사용하여 매번 다른 정보를 찾는 캐주얼 사용자와 기작성 트랜잭션을 주로 반복해서 수행하는 초보자로 구분

5) 오퍼레이터
(1) DBMS가 운영되고 있는 컴퓨터 시스템과 전산실을 관리하는 사람.
(2)

7. ANSI/SPARC 아키텍처와 데이터 독립성
1) 상용 DBMS는 1978년에 제안된 ANSI/SPARC 아키텍처
2) ANSI/SPARC 아키텍처 3단계
(1) 외부 단계 - 각 사용자의 뷰 ppt 확인
① 엔드유저와 응용 프로그래머들은 데이터베이스의 일 부분에만 관심을 가짐
② 데이터베이스의 각 사용자가 갖는 뷰
③ 여러 부류의 사용자를 위해 동일한 개념 단계로부터 다수의 서로 다른 뷰가 제공
(2) 개념 단계 - 사용자 공동체 뷰

① 조직체의 정보모델, 물리적인 구현은 고려하지 않고, 전체에 관한 스키마를 포함.
② 데이터베이스에 저장될 데이터와 데이터 간 관계, 무결성 제약조건등을 기술.
③ 데이터베이스에 대한 사용자 공동체의 뷰를 나타냄
④ 데이터베이스마다 오직 한 개의 개념 스키마가 존재.

(3) 내부 단계 - 물리적 또는 저장 뷰
① 실제 물리적인 데이터 구조에 관한 스키마
② 인덱스, 해싱 등과 같은 접근 경로, 데이터 압축 등을 기술함
③ 데이터베이스 개념 스키마에는 영향을 미치지 않으면서 성능을 향상시키기 위해 내부스키마를 변경하는 것이 바람직
④ 내부 다음 단계는 물리적 단계

(4) 스키마 간의 사상
① 외부 / 개념 사상 - 외부의 질의를 개념 단계의 스키마를 사용한 질의로 변환
② 개념 / 내부 사상 - 이를 다시 내부 단계의 스키마로 변환하여 데이터베이스 접근.

3) 데이터 독립성
(1) 상위 단계의 스키마 정의에 영향을 주지 않으면서 어떤 단계의 스키마 정의를 변경할 수 있는 것을 의미.
① 논리적 데이터 독립성 - 개념 스키마의 변화로부터 외부 스키마가 영향을 받지 않음을 의미.
② 물리적 데이터 독립성 - 내부 스키마의 변화가 개념적 스키마에 영향을 미치지 않으며, 외부 스키마에도 영향을 미치지 않는 것, (화일의 저장 구조를 바꾸거나 인덱스를 생성 및 삭제.

4) 데이터 추상화 - DBMS가 User를 위해서 데이터를 정리하여 보여주는 기능 = >데이터 추상화 기능. 외부 단계로 갈수록 추상화 수준이 올라감.

8. 데이터베이스 설계와 ER모델
1) 데이터베이스 설계
(1) 개념적 설계와 물리적 설계로 구분
(2) 개념적 설계는 실제로 DB를 djejg게 구현할 것인가와는 독립적으로 정보 사용의 모델을 개발하는 과정
(3) 물리적 데이터베이스 설계에서는 물리적인 저장 장치와 접근 방식을 다룸
(4) 개념적 데이터베이스 설계 과정에서 실세계의 엔티티, 관계, 프로세스, 무결성 제약조건 등을 나타내는 추상화 모델을 구축
(5) 엔티티는 서로 구분이 되면서 실세계에서 데이터베이스에 나타내려는 객체(사람, 장소, 사물 등)를 의미.

(6) 관계는 두 개 이상의 엔티티들 간의 연관.
(7) 프로세스는 관련된 활동
(8) 무결성 제약조건은 데이터의 정확성과 규칙

2) 개념적 수준의 모델
(1) 특정 구현 데이터 모델과 독립적으로 응용 세계를 모델링 할 수 있도록 함.
(2) 데이터베이스 구조나 스키마를 하향식으로 개발할 수 있기 위한 틀을 제공
(3) 구현 단계에서 사용되는 세가지 데이터 모델 - 관계 데이터 모델, 계층 데이터 모델, 네트워크 데이터 모델.

3) 데이터베이스 설계의 개요
(1) 현실세계의 운영과 목적을 지원하기 위해 데이터베이스를 생성하는 과정
(2) 모든 주요 응용과 사용자들이 요구하는 데이터, 데이터 간의 관계를 표현
(3) 훌륭한 데이터베이스 설계는 시간의 흐름에 따른 데이터의 모든 측면을 나타내고, 데이터 항목의 중복을 최소화하고(개념스키마), 데이터베이스에 대한 효율적인 접근을 제공하고, 데이터베이스의 무결성을 제공하고(물리적 스키마), 이해하기 쉬워야 함.

4) 데이터베이스 설계 주요 단계.
(1) 요구분석단계*
(2) 개념적 설계*
(3) DBMS의 선정
(4) 논리적 설계*
(5) 스키마 정제
(6) 물리적 설계*
(7) 보안 설계
(8) 구현단계

5) 요구사항 수집과 분석 - ERD
(1) 흔히 기존의 문서를 조사하고, 인터뷰나 설문 조사 등이 시행됨
(2) 설문 조사는 자유롭게 의견을 적어내도록 하는 방식과 주어진 질문에 대해서만 답을 하는 방식으로 구분
(3) 요구사항에 관한 지식을 기반으로 관련 있는 엔티티들과 이들의 애트리뷰트들이 무엇인가, 엔티티들의 간의 관계가 무엇인가 파악.
(4) 데이터 처리에 관한 요구사항의 전형적인 연산들은 무엇인가, 연산의 의미 접근하는 데이터의 양 등을 분석.

6) 개념적 설계
(1) 모든 물리적인 사항과 독립적으로, 현실세계에서 사용되는 정보 모델을 구축하는 과정.
(2) 사용자들의 요구사항 명세로부터 개념적 스키마가 만들어짐

(3) 높은 추상화 수준의 데이터 모델을 기반으로 정형적인 언어(그림)으로 데이터 구조를 명시함
(4) 엔티티 타입, 관계 타입, 애트리뷰트들을 식별, 애트리뷰트들의 도메인을 결정하고, 후보 키와 기본 키 애트리뷰트들을 결정.
(5) 완성된 개념적 스키마(ER 스키마)는 ER다이어그램으로 표현

7) 논리적 설계
(1) 데이터베이스 관리를 위해 선택한 DBMS의 데이터 모델을 사용하여 논리적 스키마를 생성함.
(2) 개념적 스키마에 알고리즘을 적용하여 논리적 스키마를 생성함
(3) 논리적 스키마를 나타내기 위해 관계 데이터 모델을 사용하는 경우에는 ER모델로 표현된 개념적 스키마를 관계 데이터베이스로 사상함.
(4) 관계 데이터베이스 스키마를 더 좋은 관계 데이터베이스 스키마로 변환하기 위해서 정규화 과정을 적용함.
(5) 데이터베이스 설계자가 요구사항 수집과 분석 후에 바로 논리적 설계단계로 가는 경우가 있는데, 이런 경우에는 흔히 좋은 관계 데이터베이스 스키마가 생성되지 않음.

8) 물리적 설계
(1) 처리 요구사항들을 만족시키기 위해 저장 구조와 접근 경로 등을 결정
(2) 응답시간 - 질의 갱신이 평균적으로 또는 피큰 시간 때 얼마나 오래 걸릴 것인가.
(3) 드랜잭션 처리율 - 1초당 얼마나 많은 틀내잭션들이 평균적으로 또는 피크 시간 때 처리될 수 있는가.
(4) 데이터베이스에 대한 보고서 생성에 소비하는 시간.

9) ER모델
(1) 데이터베이스 설계를 용의하게 하기 위해 1976 P.P CHEN이 제안.
(2) 현재는 eer모델이 데이터베이스 설계과정에 많이 사용되고 있음.
(3) 높은 수준으로 추상화하며, 이해하기 쉬우며, 구문들의 표현력이 뛰어나고 사람들이 응용에 대해 생각하는 방식과 가깝고, 많은 CASE도구들에서 지원됨.
(4) 실세계를 엔티티, 애트리뷰트, 엔티티들 간의 관계로 표현함
(5) 쉽게 관계 데이터 모델로 사상됨.
(6) 기본적 구문에는 엔티티, 관계, 애트리뷰트(키선정)이 있고, 기타 구문으로는 카디날리티 비율, 참여제약조건 등이 있음.
(7) 정현적 언어, 구현에 독립적이며, 데이터베이스 설계자가 엔드유저와 의사소통을 하는데 적합함.
(8) 현재는 데이터베이스 설계를 위한 다소 구형 그래픽 표기법.

10) 엔티티
(1) 명사로 표시되는, 물질명사, 추상명사

(2) 하나의 엔티티는 사람, 장소, 사물, 사건 등과 같이 독립적으로 존재하면서 고유하게 식별이 가능한 실세계의 객체
(3) 실체가 있는 것과 추상적인 것이 있음.

11) 엔티티 타입
(1) 같은 종류의 엔티티들이 많이 존재하며, 엔티티들은 엔티티 타입들로 분류됨.
(2) 엔티티 집합은 동일한 애트리뷰트들을 가진 엔티티들의 모임.
(3) 엔티티 타입은 동일한 애트리뷰트들을 가진 엔티티들의 틀.
(4) 하나의 엔티티는 한 개 이상의 엔티티 집합에 속할 수 있음.
(5) ER다이어그램에서 직사각형으로 엔티티 타입을 나타냄.

12) 강한 엔티티 타입 (그냥 엔티티 타입이라고 함)
(1) 엔티티 타입 내에서 자신의 키 애트리뷰트를 사용하여 고유하게 엔티티들을 식별할 수 있는 엔티티 타입.

13) 약한 엔티티 타입
(1) 키를 형성하기에 충분한 애트리뷰트를 갖지 못한 엔티티 타입
(2) 소유 엔티티타입이 있어야함.
(3) 소유 엔티티 타입의 키 애트리뷰트를 결합해야만 고유하게 약한 엔티티타입의 엔티티들을 식별할 수 있음.
(4) 이중 직사각형으로 표시
(5) 부분키는 점선 밑줄
(6) 부분키
① 부양 가족의 이름처럼 한 사원에 속한 부양가족 내에서는 서로 다르지만 회사 전체 사원들의 부양가족들 전체에서는 같은 경우가 생길 수 있는 애트리뷰트. (예, 아내가 같은 회사를 다닐 경우, 부양가족이 같음)

14) 애트리뷰트
(1) 하나의 엔티티는 연관된 애트리뷰트들의 집합으로 구성.
(2) 한 애트리뷰트의 도메인은 애트리뷰트가 가질 수 있는 모든 가능한 값을 의미
(3) 키 애트리뷰트는 한 애트리뷰트 또는 애트리뷰트들의 모임으로서 한 엔티티타입 내에서 각 엔티티를 고유하게 식별함.
(4) ER다이어그램에서 기본 키에 속하는 애트리뷰트는 밑줄을 그어 표시
(5) 요구사항 명세에서 명사나 형용사로 표시
(6) 애트리뷰트는 독립적인 의미를 갖지 않음.
(7) ER다이어그램에서 타원형으로 표시
(8) 애트리뷰트와 엔티티 타입은 실선으로 연결
(9) 단순 애트리뷰트
① 더 이상 다른 애트리뷰트로 나눌 수 없는(어토믹) 애트리뷰트
② ER 다이어그램에서 실선 타원으로 표현

③ ER 다이어그램에서 대부분의 애트리뷰트는 단순 애트리뷰트

(10) 복합 애트리뷰트

① 두 개 이상의 애트리뷰트로 이루어진 애트리뷰트 (주소 등등)

② 동일한 엔티티 타입이나 관계 타입에 속하는 애트리뷰트들 중에서 밀접하게 연관된 것을 모아둔 것.

(11) 단일 값 애트리뷰트

① 각 엔티티마다 정확하게 하나의 값을 갖는 애트리뷰트

② ER 다이어그램에서 단순 애트리뷰트와 동일하게 표현됨

③ ER다이어그램에서 대부분의 애트리뷰트는 단일 값 애트리뷰트

(12) 다치 애트리뷰트

① 각 엔티티마다 여러 개의 값을 가질 수 있는 애트리뷰트 (취미 등)

② ER 다이어그램에서 이중선 타원으로 표현

(13) 저장된 애트리뷰트

① 다른 애트리뷰트와 독립적으로 존재하는 애트리뷰트

② ER다이어그램에서 단순 애트리뷰트와 동일하게 표현됨

③ ER 다이어그램에서 대부분의 애트리뷰트는 저장된 애트리뷰트

(14) 유도된 애트리뷰트

① 다른 애트리뷰트 값으로부터 유추가능한 애트리뷰트

② 관계 데이터베이스에서 릴레이션의 애트리뷰트로 포함시키지 않는 것이 좋음.

③ ER 다이어그램에서 점선 타원으로 표현 (예 : 나이)

(15) 관계의 애트리뷰트

① 관계의 특징을 기술하는 애트리뷰트를 가질 수 있음. (공급 수량 등)

② 관계 타입은 키 애트리뷰트를 가지지 않음.

15) 관계와 관계 타입

(1) 관계는 엔티티들 사이에 존재하는 연관이나 연결로서 두 개 이상의 엔티티 타입들 사이의 사상으로 생각할 수 있음.

(2) 관계집합은 동질의 관계들의 집합.

(3) 관계타입은 동질의 관계들의 틀

(4) ER다이어그램에서 다이어몬드로 표기

(5) 관계 타입이 서로 연관시키는 엔티티 타입들을 관계타입에 실선으로 연결함.

16) 차수

(1) 관계로 연결된 엔티티 타입들의 개수를 의미

(2) 실세계에서 가장 흔한 관계는 두 개의 엔티티 타입을 연결하는 2진관계

17) 카디날리티

(1) 한 엔티티가 특정 관계 타입을 통해서 연관지을 수 있는 상대 엔티티 타입의 원소의 최대 수를 나타냄.

(2) 관계 타입에 참여하는 엔티티들의 가능한 조합을 제한함
(3) 관계를 흔히 1:1 1:n m:n으로 구분.
(4) 선 위에 나타냄.
(5) 카디날리티 비율 최소값 최대값은 PPT 참조 .. 이미 알고 있어서 설명하기 귀찮.

18) 전체 참여와 부분 참여
(1) 전체 참여는 어떤 관계에 엔티티 타입 예)E1의 모든 엔티티들이 관계타입 R에 의해서 어떤 엔티티 타입 E2의 어떤 엔티티와 연관되는 것을 의미
(2) 부분 참여는 위에 예에서 E1의 엔티티들이 부분적으로 E2의 엔티티와 연관 되는 것.
(3) 약한 엔티티 타입은 항상 전체참여
(4) 전체 참여는 ER 다이어그램에서 이중실선으로 표시
(5) 카디날리티 비율과 함께 참여 제약조건은 관계에 대한 중요한 제약조건

19) 다중 관계
(1) 두 엔티티 타입 사이에 두 개 이상의 관계 타입이 존재할 수 있음.

20) 순환적 관계
(1) 하나의 엔티티 타입이 동일한 관계 타입에 두 번 이상 참여하는 것.

21) 역할
(1) 관계 타입의 의미를 명확하게 하기 위해 사용됨
(2) 특히 하나의 관계 타입에 하나의 엔티티타입이 여러 번 나타나는 경우에는 반드시 역할을 표기.

22) ER 스키마를 작성하기 위한 지침?
(1) 엔티티 키는 애트리뷰트 이외에 설명 정보를 추가로 가짐
(2) 다치 애트리뷰트는 엔티티로 분류해야 함
(3) 애트리뷰트들이 직접적으로 설명하는 엔티티에 에트리뷰트들을 붙임
(4) 가능한 복합 식별자를 피함
(5) 관계는 일반적으로 독자적으로 존재할 수 없지만 엔티티 타입과 과녜타입을 절대적으로 구분하는 것은 어려움.

23) 데이터베이스 설계 순서
(1) 요구사항 수집
(2) 엔티티 타입 식별
(3) 관계 타입 식별
(4) 관계의 카디날리티 비율 설정
(5) 엔티티 타입과 관계 타입들에 필요한 애트리뷰트들을 식별하고, 각 애트리뷰트가 가질 수 있는 값들의 집합을 식별.
(6) 엔티티 타입들을 위한 기본 키를 식별
(7) 응용을 위한 ER스키마 다이어그램을 그림

(8) ER스키마 다이어그램이 응용에 대한 요구사항과 부합되는지 검사

(9) ER스키마 다이어그램을 DBMS에서 사용되는 데이터베이스 모델로 변환.

24) ER 모델의 또 다른 표기법

(1) 등등은 피피티 참조하자 .

9. 관계 데이터 모델과 제약조건

1) 가장 개념이 다눈한 구현 데이터 모델 중 하나.

2) 1970년 IBM연구소 E.F. Codd가 제안

3) 최초로 구현한 가장 중요한 관계 DBMS시제품은 1970 IBM연구소에서 개발된 System R

4) 성공을 거둔 요인**어렵

(1) 바탕이 되는 데이터 구조로써 간단한 테이블을 사용.

(2) 중첩된 복잡한 구조가 없음.

(3) 집합 위주로 데이터를 처리

(4) 다른 데이터 모델에 비해 이론이 잘 정립되어 있음.

(5) 숙련되지 않은 사용자도 쉽게 이해할 수 있음.

(6) 표준 데이터베이스 응용에 대해 좋은 성능을 보임

(7) 관계 데이터베이스 설계와 효율적인 질의 처리 면에서 뛰어난 장점을 가짐.

10. 관계 데이터 모델

1) 모든 데이터를 논리적으로 동일한 구조(릴레이션)의 관점에서 구성

2) 논리적으로 연관된 데이터를 연결하기 위해서 링크나 포인터를 사용하지 않음.

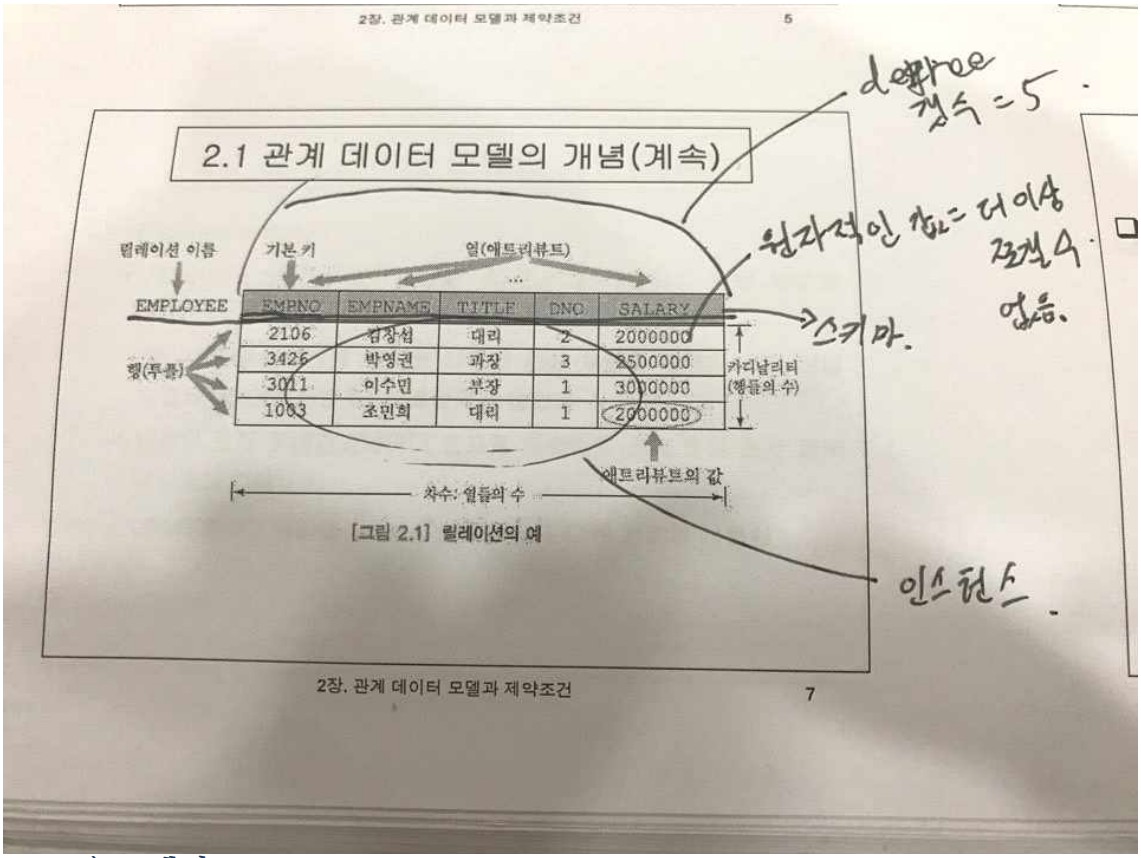
3) 응요 프로그램들은 데이터베이스 내의 레코드들의 어떠한 순서와도 무관하게 작성됨.

4) 선언적인 질의어를 통한 데이터 접근을 제공

5) 사용자는 원하는 데이터만 명시하고, 어떻게 이 데이터를 찾을 것인가는 DBMS가 알아서 함.

11. 기본용어

1) 릴레이션 - 2차원의 테이블
2) 레코드 - 릴레이션의 각 행
3) 투플 - 레코드를 좀 더 공식적으로 부르는 용어
4) 애트리뷰트 - 릴레이션에서 이름을 가진 하나의 열



5) 도메인
(1) 한 애트리뷰트에 나타날 수 있는 유효 값들의 집합
(2) 각 애트리뷰트의 도메인의 값들은 원자 값
(3) 프로그래밍 언어의 데이터 타입과 유사함
(4) 동일한 도메인이 여러 애트리뷰트에서 사용될 수 있음.
(5) 복합 애트리뷰트나 다치 애트리뷰트는 허용되지 않음
(6) 도메인 정의 예).

```
CREATE DOMAIN EMPNAME CHAR(10)
CREATE DOMAIN EMPNO INTEGER
CREATE DOMAIN DNO INTEGER
```

6) 차수와 카디날리티
(1) 차수 - 릴레이션에 들어있는 애트리뷰트들의 수, 유효한 릴레이션의 최소 차수는 1
(2) 카디날리티 - 유효한 릴레이션은 카디날리티 0을 가질 수 있음. 릴레이션의 카디날리티는 시간이 지남에 따라 계속 변함

7) 널값

(1) '알려지지 않음' 또는 '적용할 수 없음'을 나타내기 위해 널값을 사용.
(2) 예 : 사원 릴레이션에 새로운 사원에 관한 투플을 입력하는데, 신입사원의 DNO(부서번호)가 결정되지 않았을 수 있음.
(3) 널값은 숫자 도메인의 0 이나 문자열 도메인의 공백 문자 또는 공백 문자열과 다름
(4) DBMS들마다 널값을 나타내기 위해 서로 다른 기호를 사용함.

8) 릴레이션 스키마
(1) 릴레이션을 구성하는 틀로서 릴레이션의 이름과 릴레이션의 애트리뷰트들의 집합으로 표현
(2) 표기 - 릴레이션이름 (애트리뷰트1, 애트리뷰트2, 애트리뷰트N)
(3) 기본 키 애트리뷰트에는 밑줄 표시

9) 릴레이션 특성
(1) 각 릴레이션은 하나의 레코드 타입만 포함.
(2) 한 애트리뷰트의 값은 모두 같은 도매인 값.
(3) 애트리뷰트 순서는 중요하지 않음.
(4) 동일한 투플이 두 개 이상 존재 안함.
(5) 투플의 각 애트리부트는 원자값만 가짐.
(6) 애트리부트 이름은 한 릴레이션 내에서만 고유

10) 릴레이션의 키
(1) 각 투플들을 고유하게 식별할 수 있는 하나 이상의 애트리뷰트들의 모임
(2) 수퍼키
① 한 릴레이션 내의 특정 투플을 고유하게 식별하는 하나의 애트리뷰트 또는 애트리뷰트들의 집합.
(3) 후보키
① 각 투플을 고유하게 식별하는 최소한의 애트리뷰트들의 모임 - 즉 기본키의 후보
② 모든 릴레이션에는 최소 한 개 이상의 후보 키가 있음.
③ 두 개 이상의 애트리뷰트로 이루어질 수 있으며 이런 경우 복합 키라고 부름
(4) 기본키
① 한 리레이션에 후보 키가 두 개 이상 있으면 설계자가 이들 중에서 하나를 기본키로 선정함.
(5) 대체키
① 기본키가 아닌 후보키
(6) 외래키
① 어떤 릴레이션의 기본 키를 참조하는 애트리뷰트
② 관계 데이터베이스에서 릴레이션 간의 관계를 나타내기 위해 사용
③ 외래 키 애트리뷰트는 참조되는 릴레이션의 기본 키와 동일한 도메인을 가져야

함.

④ 자신이 속한 릴레이션의 기본 키의 구성요소가 되거나 되지 않을 수 있음.

12. 데이터 무결성 제약조건

1) 데이터 정확성 혹은 유효성을 의미
2) 규칙들을 묵시적 혹은 명시적으로 정의
3) 데이터베이스가 갱신될 때 DBMS가 자동적으로 일관성 조건을 검사함.
4) 도메인 제약조건
   (1) 애트리뷰트 값이 원자값 이어야 함.
   (2) 애트리뷰트 값의 디폴트 값, 가능한 값들의 범위를 지정
   (3) 데이터 형식을 통해 값들의 유형을 제한, CHECK 제약 조건을 통해 값들의 범위를 제한.
5) 키 제약조건
   (1) 키 애트리뷰트에 중복된 값이 존재해서는 안됨
6) 기본 키와 엔티티 무결성 제약조건
   (1) 릴레이션 의 기본키를 구성하는 애트리뷰트들은 널값을 가질 수 없음
   (2) 대체 키에는 적용 안됨.
7) 외래 키와 참조 무결성 제약조건
   (1) 참조 무결성 제약조건은 두 릴레이션의 연관된 투플들 사이의 일관성을 유지하는데 사용됨
   (2) 관계 데이터베이스가 릴레이션들로만 이루어지고 릴레이션 사이의 관계들이 다른 릴레이션의 기본 키를 참조하는 것을 기반으로 하여 묵시적으로 표현되기 때문에 외래키의 개념이 중요.
   (3) 릴레이션 R2의 외래 키가 릴레이션 R1의 기본 키를 참조할 때 참조 무결성 제약조건은 아래의 두 조건 중 하나가 성립되면 만족.
      ① 외래 키의 값은 R1의 어떤 투플의 기본 키 값과 같다.
      ② 외래 키가 자신을 포함하고 있는 릴레이션의 기본 키를 구성하고 있지 않으면 널값을 가진다.
8) 무결성 제약조건의 유지
   (1) 데이터베이스 갱신 연산은, 삽입, 삭제, 수정연산으로 구분함.
   (2) DBMS는 외래 키가 갱신되거나, 참조된 기본 키가 갱신되었을 때, 참조무결성 제약조건이 위배되지 않도록 해야함.
   (3) 삽입
      ① 참조되는 릴레이션에 새로운 투플이 삽입되면 참조 무결성 제약조건은 위배되지 않음.
      ② 참조되는 릴레이션에새로 삽입되는 투플의 기본 키 애트리뷰트 값에 따라서는, 도메인, 키 제약조건, 엔티티 무결성 제약조건 등을 위배할 수 있음.

③ 참조하는 릴레이션에 새로운 투플을 삽입할 때에는 4가지 제약조건이 위배될 수 있음.

(4) 삭제

① 참조하는 릴레이션에서 투플이 삭제되면 모든 제약조건이 위배되지 않음.

② 참조되는 릴레이션에서 투플이 삭제되면 참조 무결성 제약조건을 위배하는 경우가 생길 수도 있음.

(5) 참조 무결성 제약조건을 만족시키기 위해서 DBMS가 제공하는 옵션

① 제한 - 위배를 한 연산을 거절

② 연쇄 - 참조되는 릴레이션에서 투플을 삭제하고 참조하는 릴레이션에서 투플을 참조하는 투플들도 삭제.

③ 널값 - 투플이 삭제되면 널값이 들어감.

④ 디폴트 - 널값을 넣는 대신 디폴트 값을 넣는다는 거 빼곤 차이없음.

(6) 수정

① DBMS는 수정하는 애트리뷰트가 외래 키인지 검사.

② 수정하려는 애트리뷰트가 기본 키도 아니고 외래키도 아니면 수정이 참조 무결성 제약조건을 위배 안함.

③ 기본 키나 외래 키를 수정하는 것은 하나의 투플을 삭제하고 새로운 투플을 그 자리에 삽입하는 것과 유사하므로 삽입 및 삭제에서 설명한 제한, 연쇄, 널값, 디폴트값 규칙이 수정에도 적용함.

13. ER스키마를 관계 모델의 릴레이션으로 사상.

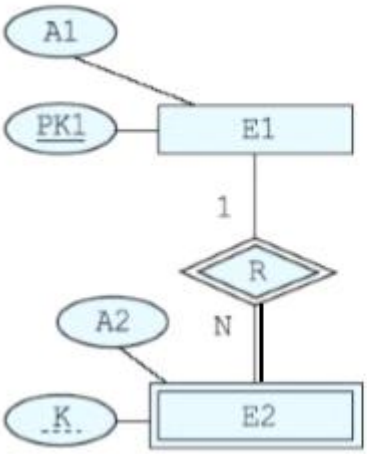
1) ER관계 사상 알고리즘.

2) 단계 1: 정규 엔티티 타입과 단일 값 애트리뷰트

[그림 5.44] 정규 엔티티 타입을 릴레이션으로 사상

3) 단계 2: 약한 엔티티 타입과 단일 값 애트리뷰트

4) 단계 3: 2진 1:1 관계 타입



E1

PK1	A1

E2

PK1	K	A2

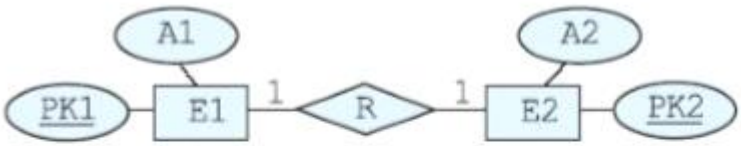
(1) ER 스키마의 각 2진 1:1 관계 타입 R에 대하여, R에 참여하는 엔티티 타입에 대응되는 릴레이션 S와 T를 찾음

(2) S와 T 중에서 관계 타입에 완전하게 참여하는 릴레이션을 S의 역할을 하는 릴레이션으로 선택함

(3) T의 기본 키를 S 릴레이션에 외래 키로 포함시킴

(4) 관계 타입 R이 가지고 있는 모든 단순 애트리뷰트(복합 애트리뷰트를 갖고 있는 경우에는 복합 애트리뷰트를 구성하는 단순 애트리뷰트)들을 S에 대응되는 릴레이션에 포함시킴

(5) 두 엔티티 타입이 관계 타입 R에 완전하게 참여할 때는 두 엔티티 타입과 관계 타입을 하나의 릴레이션으로 합치는 방법도 가능함



방법 1 :

E1

PK1	A1
K1	

E2

PK2	A2	FK1
K2		K1

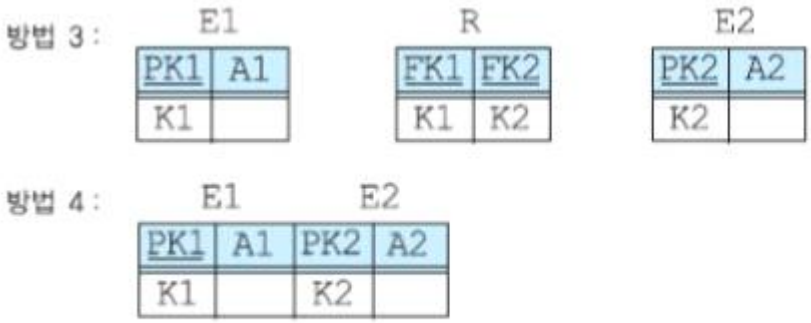
방법 2 :

E1

PK1	A1	FK2
K1		K2

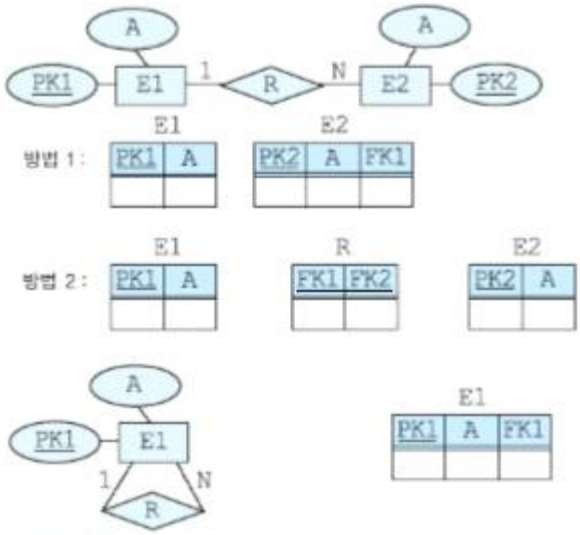
E2

PK2	A2
K2	



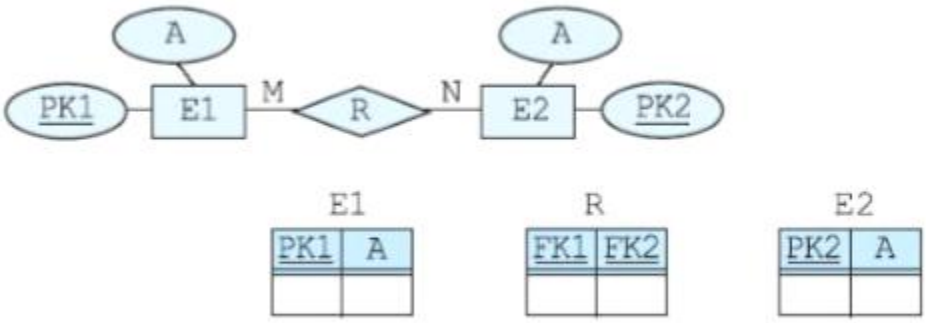
[그림 5.46] 2진 1:1 관계 타입을 릴레이션으로 사상

5) 단계 4: 정규 2진 1:N 관계 타입



[그림 5.47] 정규 2진 1:N 관계 타입을 릴레이션으로 사상

6) 단계 5: 2진 M:N 관계 타입



[그림 5.48] 2진 M:N 관계 타입을 릴레이션으로 사상

7) 단계 7: 다치 애트리뷰트

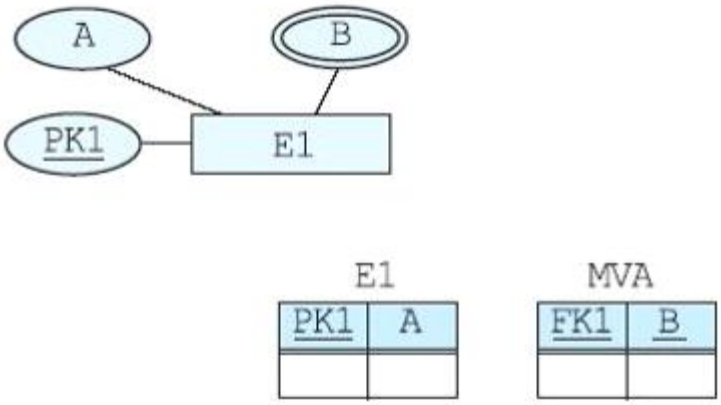


그림 5.50] 다치 애트리뷰트를 릴레이션으로 사상

8) 단계 2: 약한 엔티티 타입과 단일 값 애트리뷰트 DEPENDENT(Empno, Depname, Sex
9) 단계 3: 2진 1:1 관계 타입 PROJECT(Projno, Projname, Budget, StartDate, Manager)
10) 단계 4: 정규 2진 1:N 관계 타입 EMPLOYEE(Empno, Empname, Title, City, Ku, Dong, Salary, Dno) PART(Partno, Partname, Price, Subpartno
11) 단계 5: 2진 M:N 관계 타입 WORKS_FOR(Empno, Projno, Duration, Responsibility)
12) 단계 7: 다치 애트리뷰트 PROJ_LOC(Projno, Location)

14. SQL 개요
1) SQL은 비절차적 언어이므로 사용자는 자신이 원하는 바를 명시하며, 원하는 것을 처리하는 방법은 명시할 수 없음.
2) 관계 DBMS는 사용자가 입력한 SQL문을 번역하여 사용자가 요구한 데이터를 찾는데 필요한 모든 과정을 담당.
3) SQL 구성요소
(1) 데이터 정의어
① CREATE -(DOMAIN, TABLE, VIEW, INDEX)
② ALTER - (TABLE)
③ DROP - (DOMAIN, TABLE, VIEW, INDEX)
④ CREATE SCHEMA MY_DB AUTHORIZATION(권한 부여) KIM;
⑤ DROP SCHEMA MY_DB RESTRICT;
⑥ DROP SCHEMA MY_DB CASCADE;
⑦

```
CRATE TABLE DEPARTMENT
```

```
(DEPTNO INTEGER NOT NULL;
DEPTNAME CHAR(10),
FLOOR INTEGER,
PRIMARY KEY(DEPTNO));
```

```
CREATE TABLE EMPLOYEE
(EMPNO INTEGER NOT NULL,
EMPNAME CHAR(10),
TITLE CHAR(10),
MANAGER INTEGER,
SALARY INTEGER,
DNO INTEGER,
PRIMARY KEY(EMPNO),
FOREIGN KEY(MANAGER) REFERENCES EMPLOYEE(EMPNO),
FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DEPTNO));
```

```
릴레이션 제거
DROP TABLE DEPARTMENT;
테이블 변경
ALTER TABLE EMPLOYEE ADD PHONE CHAR(13);
인덱스 생성
CREATE INDEX EMPDNO_IDX ON EMPLOYEE(DNO);
도메인 생성
CREATE DOMAIN DEPTNAME CHAR(10) DEFAULT '개발';
```

(2) 데이터 조작어

① **UPDATE** DEPARTMENT, **SET** FLOOR=10, **WHERE** DEPTNO = 1;

② **DELETE FROM** DEPARTMENT **WHERE** DEPTNAME = 총무;

③ **INSERT INTO** DEPARTMENT **VALUES**(5, '연구', 9);

④ **SELECT** DEPTNAME, FLOOR, **FROM** DEPARTMENT, **WHERE** DEPTNO = 1 OR DEPTNO = 3;

(3) 데이터 제어어

```
제약조건
CREATE TABLE EMPLOYEE
(EMPNO INTEGER NOT NULL,
EMPNAME CHAR(10) UNIQUE,
TITLE CHAR(10) DEFAULT '사원'
MANAGER INTEGER,
```

```
SALARY INTEGER CHECK (SALARY < 6000000),
DNO INTEGER CHECK (DNO IN(1,2,3,4)) DEFAULT 1,
PRIMARY KEY (EMPNO)
FOREIGN KEY (MANAGER) REFERENCES EMPLOYEE(EMPNO),
FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DEPTNO)
ON DELETE SET DEFAULT ON UPDATE CASCADE);
```