Chap 2 High Level DataBase Models

소개 (데이터 모델 : 데이터와 데이터들 간의 관계를 기술하는 지명적 도구, DB의 논리적 구조를 명시)

* E/R 모델의 창안 : DB를 만들어야 겠다는 Ideas 에서 출발
  - 어떤 정보를 저장, 정보들끼리 어떤 연관성을 지니는가 어떤 제약조건을 가질수 있는가
* E/R 모델의 목적 : High-Level-Design (개념설계)
* DB 디자인 : 사람들이 알아보기 쉽게 디자인을 했다. DB의 논리적 구조와 그에 대한 제약조건정

Ideas → High Level Design (개념적 설계) → convert → Relational DB Schema (구현설계) → Relational DBMS

* DB 설계를 위한 관계형 모델 (relational model)
  - 관계형 모델은 실제 상황을 모두 설명할수있는 여러 개념보다는 하나의 개념만을 가지고 있다.
  - 간단하게 표현하여 DB의 효율적인 구현이 관계모델의 장점 → 다양한 개념을 표현하기엔 부족할수있다.

E/R 모델

* E-R 모델 : 데이터의 구조가 그래픽 (구체적, 체계적으로 시각화) 으로 표현되었다.
  - 스키마 : DB의 구조 기술
  - 인스턴스 : 스키마의 특정값들의 내용
* E/R 다이어그램 : DB의 구조를 기술하는 표기법, DB의 스키마를 설명하기 위한 표현 방법
  - 객체 : 엔티티 (▭) 동등으로 존재하는 객체이며 종속된 객체는 존재하지 않는다.
  - 속성 : 객체의 속성 (⬭) 객체가 가지는 속성을 의미
  - 관계 : 둘 이상 Table 간의 연결 (◇) 엔티티 집합들 간의 관계를 의미한다.

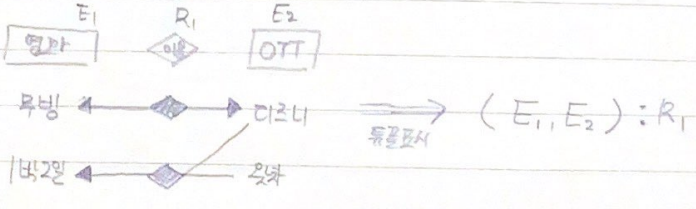
⇒ 튜플의 구성요소는 속성(attributes)이 아닌 관계도 아닌 엔티티(Entity) 이다.
(튜플 = tuples)

* E/R 관계성의 다중연관성
  - n:1 관계 ▭—◇→▭ " → " 화살표는 가리키는 방향으로 최대 1개의 객체를 가리킨다.
  - 1:1 관계 ▭←◇→▭
  - n:n 관계 ▭—◇—▭

(ex)

영화	OTT
무빙	디즈니
1박2일	넷플
1박2일	왓챠

E/R model



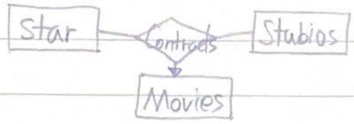
튜플표시 → $(E_1, E_2) : R_1$

"왓챠는 1개의 1박2일을 가지고있다"

→ 세개이상의 엔티티 집합들이 참여하는 관계성.

* 복잡한 (3개 이상의 객체) 관계 / 다중방향 관계성

(ex) 스튜디오가 특정스타와 계약하여 특정 영화에 출연하는 경우.



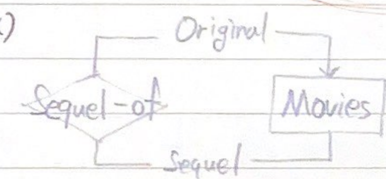
⇒ 튜플 (Sta . Stu . Mov)

순서상관 X

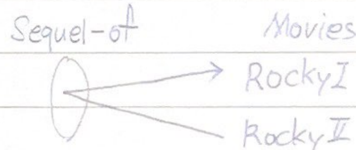
* 화살표의 의미 : 관계에 있는 다른 개체 집합들 각각에서 하나의 개체를 선택한다면 그 인스턴스들은 (그 객체는) 엔티티 셋 (다른 객체와)에서 많아도 하나와의 관계가 있다.

* 관계에서의 역할(성) : 자기 자신과의 관계, 우속각의 관계를 표현, 한 엔티티 집합이 한 관계성에서 두번이상 사용가능.

(ex)



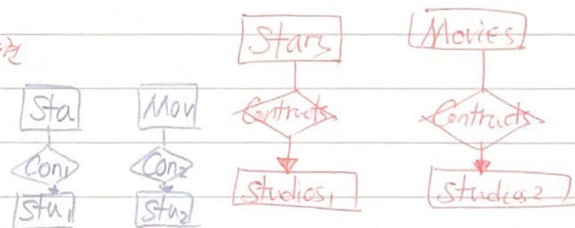
관계에서 용라 (Original . Sequel)도 적어야함



Rocky II is a Sequel-of Rocky I

* 복잡한 관계 ——— 다진관계로 표현 →

⇒



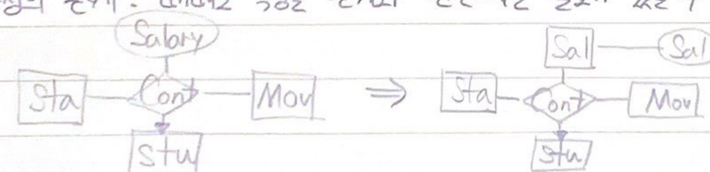
(ex) Movies 는 하나의 Studios$_1$ (회사$_1$)를 가집니다.

Stars 는 하나의 Studios$_2$ (회사$_2$)를 가집니다.

⇒ 튜플: (Studio1, Stu2, Sta, Mov)

(순서상관 X , 관계표시 X)

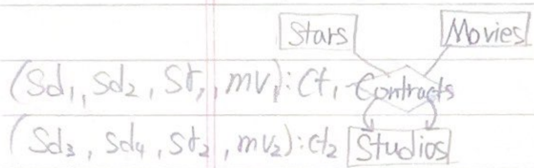
* 속성의 관계 : 때때로 속성을 관계와 연관 지을 필요가 있을 수 있다. (관계성이 있는 애트리뷰트)



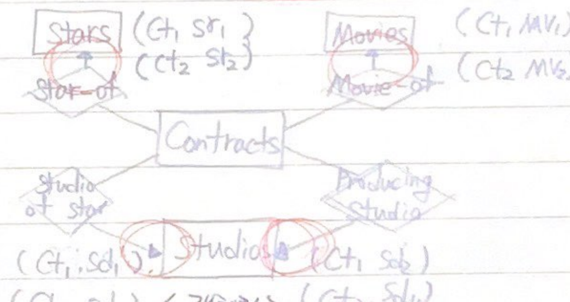
* 다중관계를 이진관계로 변환

- 일부데이터 모델은 관계를 이진으로 제한
- UML & ODL을 사용하여 변환
- 연결개체 (관계의 단일 인스턴스를 나타내는 개체) 생성 → 다중관계제거
- 1 : 다 관계 생성

(ex)



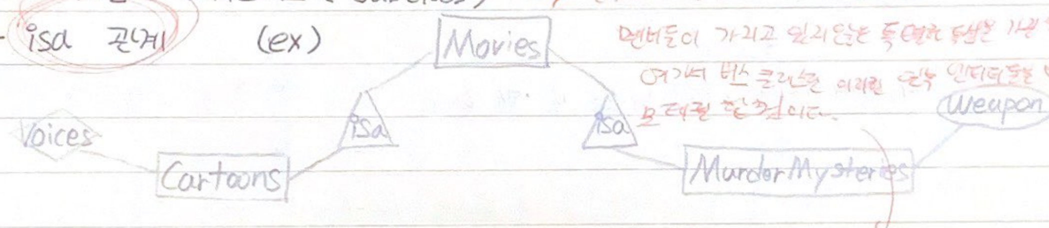
<관계형>



<객체형>

* E/R 모델의 하위클래스 (Subclass) ⇒ 일부엔티티들이 그 엔티티 집합의 모든 멤버들이 가지고 있지 않은 특별한 특성을 가질 수 있다. 여기서 서브클래스는 이러한 일부 엔티티들을 나타내고 모델링 할 것이다.

- isa 관계 (ex)



클래스 — 서브클래스 관계시 그리고 삼각형으로 표현하며 1:1 관계를 가진다.

- 영화의 하위분류는 (isa) 만화와 살인미스테리 이다.
- 두 개의 하위분류를 다 포함할수 있다. (ex) 전쟁 애니
- 하나의 분류만 해당될수 있다 (ex) 전쟁영화

◎ 설계원칙 (Design - Principles)

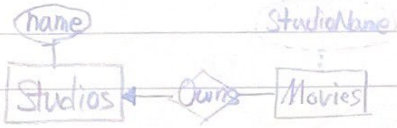
* 충실성 다중연관성
* 중복회피 비슷한 엔티티 제거
* 단순화 필요없는것 제거
* 올바른 관계성의 선택 관계의 필요유무
* 올바른 요소의 선택 객체, 혹은 애트리뷰트 를 판단 (엔티티)

* 충실성 (Faithfulness) : 관계, 다중관계에서 중요

- 설계는 다루고자 하는 상황을 충실하게 나타내야 함.
- DB에 유지, 관리하려는 정보만 모델링, 모델링 되는 현실을 정확히 반영해야 함.

(ex) Star와 Movies 사이의 관계는 다:다 가 적절하다.

* 중복회피 (Avoiding redundancy) (ex)



- 공간이 낭비가 될 수 있는 객체, 관계, 변수, Class 등을 제거
- 여분의 공간이 필요하며 (ex 업데이트등 때문에) 이게 없다면 오류가 발생할 수 있다.

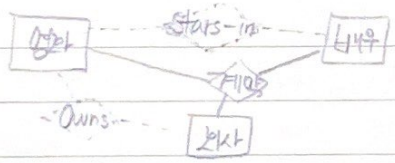
* 단순화

- 절대적인것 이상의, 필요이상의 복잡한 설계에 빠지 않는다.

* 올바른 관계성의 선택

- 개체들의 관계에 의해 다양한 방식으로 연결될수 있지만, 가능한 모든 관계를 추가하는 것은 좋은 좋은 생각은 아니다.
- 같은 정보를 다른 정보에서 추출할수 있다면 중복이 없는 관계를 선택
- 인터티는 다양한 방식의 관계를 생성할수 있으나 이것이 바람직한것은 아니다.

(ex)

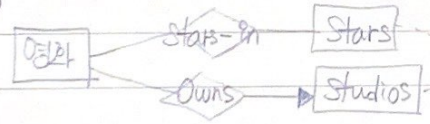


⇒ 3개의 객체를 Contracts 로 묶을수 있다

- 이경우 되면 Stars in, Owns 는 필요여부에 따라 삭제된다.
- 회사에 영화는 관계 되지만 배우가 없다면 Own 관계는 살려야한다.

→ 영화에 대해 계약이 계속있으면 Own 삭제

(ex)

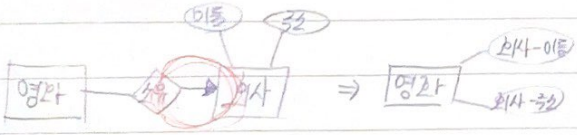


⇒ 만약 works-for 라는 관계가 필요하다면 works-for 라는 관계는 Stars-in & Owns 관계와 독립적이라는 것이다

* 올바른 모델의 선택

- 실세계의 개념을 표현하기위해 여러 설계타입을 사용할 수 있으며 무엇을 사용하는지 선택해야한다.
- 다음의 조건에서 객체 대신 속성을 선호하며, 꼭 속성으로 표시할 필요는 없다. (가급적이면 좋은것이다)
  - 객체와 관련된 모든 관계에 대해 객체는 다:1 관계에서 1의 객체여야 한다.
  - E(객체)의 속성은 다른 속성에 의존하지 않는다.
  - 어떤 관계도 E(객체)와 두번 이상의 관련이 없다.

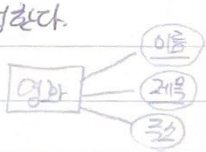
(ex)



- Studios 의 속성을 영화의 속성으로 만든다.
- 회사의 단순이름정도만 필요하다면 추가의 경우가 너무많아지므로 회사의 더많은 정보를 가지고, 사용한다면 전자가 바람직하다.
- 1대다 관계를 생각하지않고 회사에 해당하는 영화를 삭제하게 되면 "삭제 오류" 발생

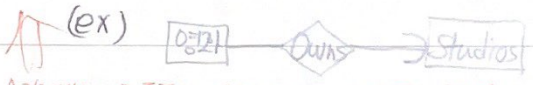
제약조건 (Constrains in the EIR Model)

* 키

- 키는 여러개가 존재 할수 있지만, 기본키 (primary key)를 하나 설정해야 한다.
- key는 E(객체)의 값이다.
- 식별할수 있는 (중복 x, 누락 x) 값을 K로 설정한다.
- key는 밑줄을 그어 표시한다. (ex) 

≠ ISA 계층구조에서 키를 구성하는 모든 어트리뷰트는 모두 루트엔티티 집합에 존재

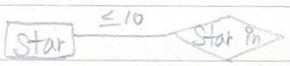
* 참조무결성 : 관계성에서 참조되는 엔티티는 반드시 존재해야한다.

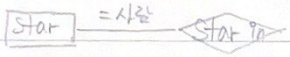
- 생기게 된 배경 (카드, 계좌 : 계좌번호를 제거하면 자동이체 되야하는 금액이 무엇인지 생기면서 계속 연체금이 쌓이는 오류가 생성된다)
- 한 Context 에서 나오는 값은 다른 Context 에서 나타나야 한다.
- 둥근 화살표로 표시한다. (ex) 

A릴레이션의 B튜플의 C릴레이션의 D튜플은 참조한다. 할수도 B튜플은 C릴레이션 내에 존재해야한다.

* 차수제한 (도메인 제한)

- 관계성에 참여하는 엔티티를 제한

(ex) 10명의 영화만 

사람만 

화살표 ⟼ 는 ≤1 사용

참조무결성은 =1 사용

* 단일값 제약

- 중복된 값들이 가면 안된다 ⇒ 키 (값이 반드시 존재), 다른 어트리뷰트 (NULL값 사용가능)
- E ⟼ F 에서 E에 있는 엔티티 e는 단일값 제약

* 일반제약

무결성제약조건 : 현실세계에 저장되는 데이터를 제한하는 조건

도메인 제약조건 : 도메인에 속하는 값이어야 한다.

엔티티 무결성제약조건 : 기본키 값은 NULL X

참조무결성 제약조건 : 관계성에서 참조되는 엔티티는 반드시 존재해야한다.

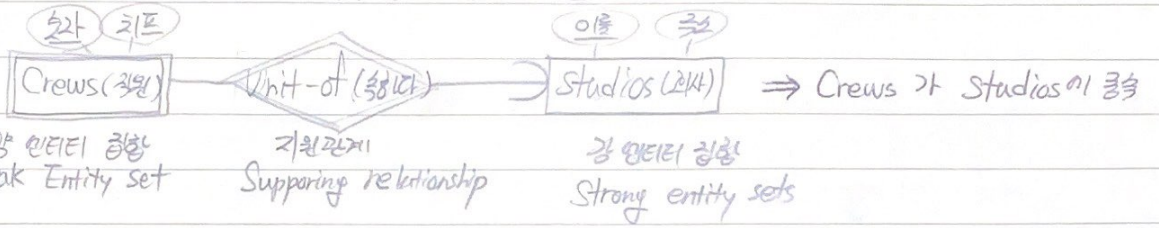
A튜플이 B튜플을 참조하려면 A튜플이 B릴레이션에 내에 존재해야한다

키 제약조건 : 서로다른 튜플은 동일한 키값은 가지지 않아야 한다는 조건

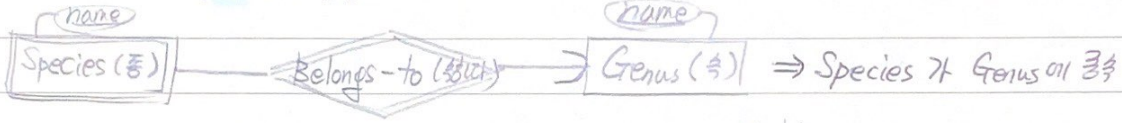
약한 객체 집합 (Weak Entity Sets)

- 키가 다른 엔티티 집합에 속하는 속성으로 구성된 엔티티 집합.
- 일부분을 기준으로 계층의 하위 단위인 엔티티 집합 , 키의 일부또는 전부가 다른 엔티티 집합에 있는 어트리뷰트로 구성되는 집합

(ex) 한 영화스튜디오에는 여러명의 영화제작진이 있습니다.



(ex) 종은 하나의 속에 하위 단위이다



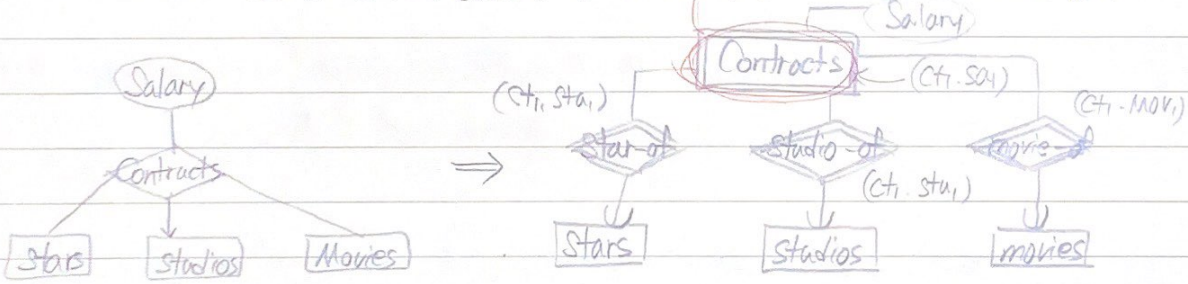
(Homo Sapiens)

- 종속적인 느낌 (≒ Weak)
- 약 엔티티를 사용하면 참조무결성이 자동생성
- Crew는 생략할 수 있지만 Studios는 Crews 때문에 상세히 힘들다.
- 1:다 같은 경우 화살표를 표기해야하지만 아직 구조상 그렇게 자세하게 표기할 필요가없으므로 넘어감.

* 

- W는 S가 없으면 존재하지 않는다. 즉 W는 혼자 지울수 있지만 S는 그냥 지울수 없다.

Star와 studios.movies의 키가 모두 모여있는 엔티티이다.



- Contracts 는 상세 가능 (Ct1)
- Stars . studios . Movies 를 상세하려면 관계를 정리한후 처리 가능
-