**- Chapter 07: Database Design & The E-R Model**

**- Creating a Database**

필요를 고려해야 함. – 필요한 Reports, 만들고 싶은 Inquiries.

각각 column에 어떤 데이터가 필요한지.

Field의 특징을 정해야 함.

Field name: 각각 field는 유일한 이름을 가져야 함.

Field type & length: character, numeric, date…

테이블을 만들 때: 테이블의 각각의 field 정의, primary key를 정의.

**- Entering the Data**

Data를 datasheet나 그래픽 형태로 넣을 수 있음.

**- Exercise: Part-Project-Supplier DB**

**Parts** are supplied to **projects**. There can be multiple **suppliers** for a part. Each part has **part-id, part-name, and size**. You must keep track of the **project managers** and **contact information** for each project.

**- Your Design**

**- Design 1**

가장 간단한 예. 하나의 테이블에 모두 넣기.

**- Design 1.2**

1에다가 S-contact도 추가.

DB 설계에서 가장 피해야 할 것이 중복. Redundancy.

중복이 나쁜 이유? Inconsistency를 일으킬 수 있는 여지를 제공.

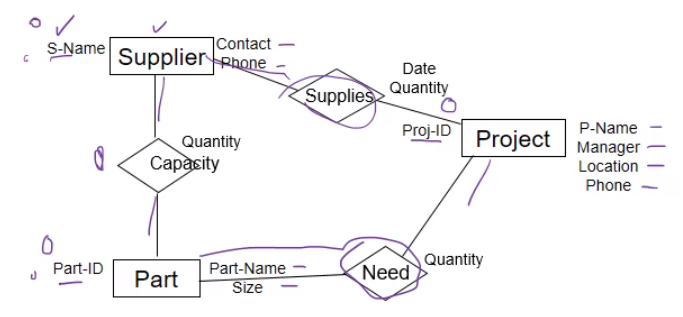
이 설계에서는 많은 중복이 생김.

**- Design 2**

테이블을 Supplier, Project, Part로 분리함.

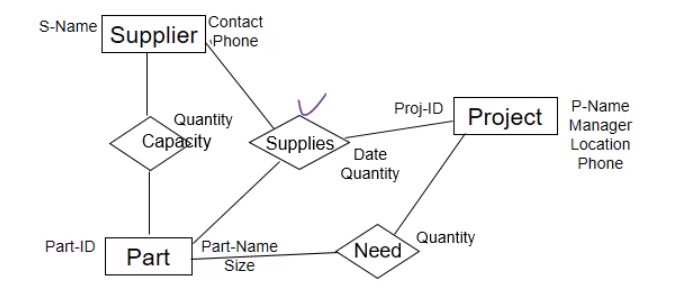
이 디자인의 문제는, 각각의 테이블의 정보가 연결되지 않음.

**- Design 3 <ER Diagram>**

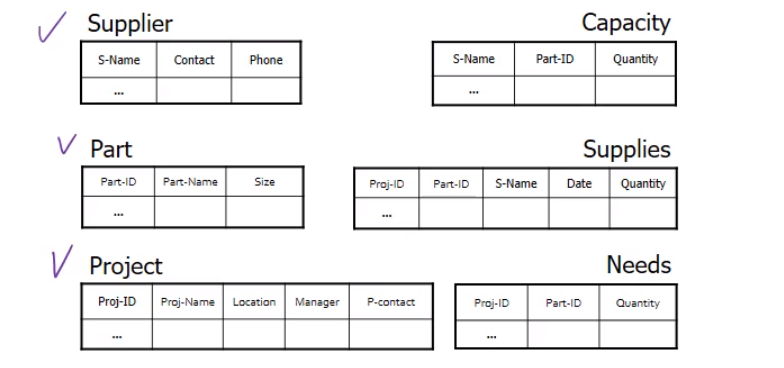


각각의 entity들과 그 relationship을 정의.

이 때 Supplies에서 supplier가 무엇을 공급했는지 모름. 아래는 수정된 관계.



**- Design 3 <Relations>**



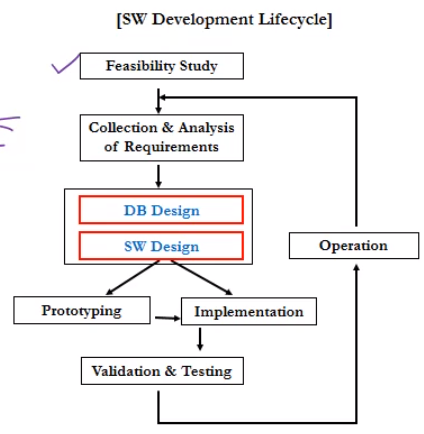
위와 같이 6개의 테이블을 만들 수 있음.

**- Database Design**

다음과 같은 DB schema를 결정해야 함.

모든 정보를 담고, 최소한의(혹은 아예 없는) redundancy를 가지고, 효율적이고 효과적인 데이터 연산들을 허용해야 함.

SW 시스템의 연산과 유지보수의 비용을 줄이는 데 아주 중요함.



**- Phases of Database Design**

3개의 phase가 있음.

1. Conceptual Design

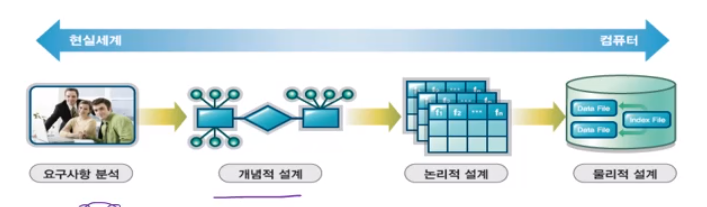
ER schema를 정의하는 부분. 유저 요구의 최적화된 description을 제공하기 위함.

1. Logical Design

DBMS에 data model을 구현하는 것. Relation schema 등을 정의하는 것. 보통 ER모델 -> Relational 모델로 나타냄.

1. Physical Design

Database의 물리적인 특성들을 정의하는 단계. (performance 이슈를 중요하게 봄. Index, sequential order 등)



**- Entity-Relationship Model**

1976년 P.Chen에 의해 제안됨.

데이터베이스를 디자인하는 데 매우 효과적인 툴.

간단한 모델, 유저와 디자이너와 구현하는 사람들 사이에 의사소통의 효과적인 수단.

E-R 모델은 implementation model 은 아님.

내부 구조가 E-R 모델로 이루어져 있는 DBMS는 없음.

**- Database Modeling**

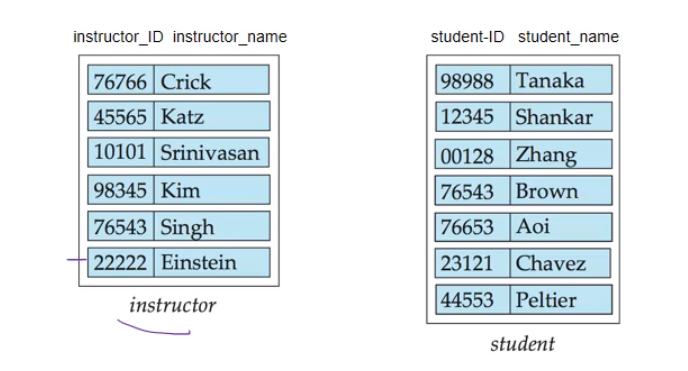
데이터베이스는 entity의 collection, 그리고 entity들의 관계로 나타낼 수 있음.

Entity: 다른 object들과 구분되는 object. Entity instance의 예시: specific person, company, event.

Attribute: entity는 attribute를 가짐. Ex) 사람들은 이름과 주소를 가짐.

Entity Set: 같은 property를 공유하는 같은 타입의 entity의 집합. Ex) 모든 사람의 집합 , 모든 회사의 집합, 모든 트리의 집합, 휴일의 집합 등..

**- Entity & Entity Sets – examples**

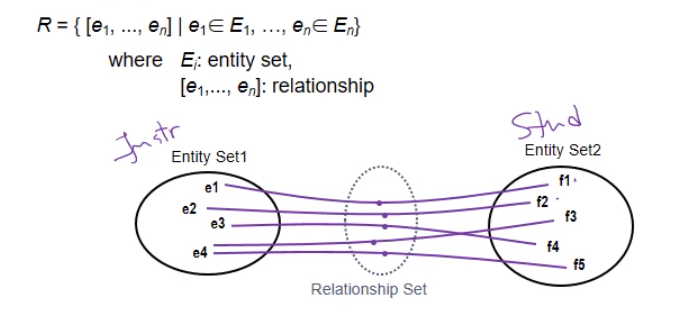


위에서 각각의 instructor entity가 모여서 instructor entity set이 됨. Student도 마찬가지.

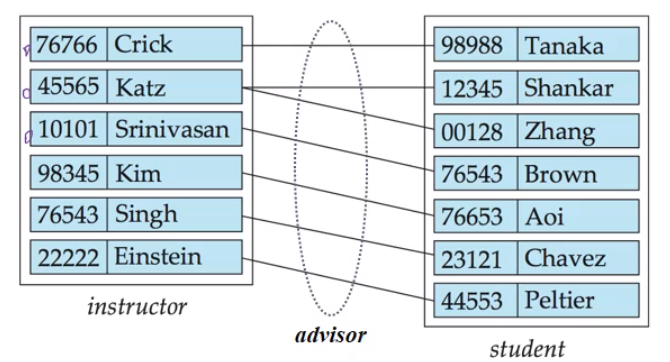
**- Relationships**

Relationship은 entity들 사이에서 정의된다.

Relationship set은 아래와 같음.



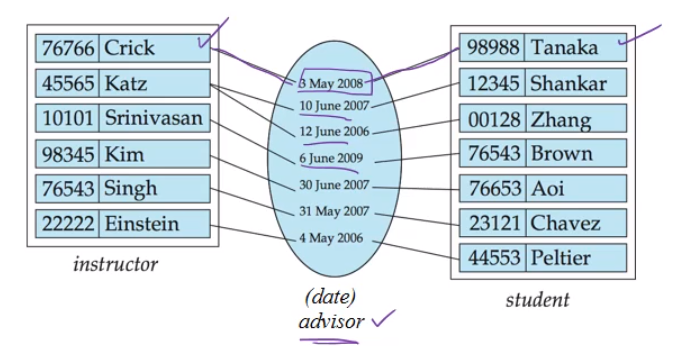
**- Relationship Set advisor**



위와 같이 각 entity들 사이의 relation들의 집합.

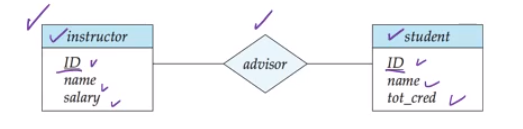
**- Attributes of Relationships**

Relationship 또한 attribute를 가질 수 있음.



위의 예시에서 advisor relationship은 date라는 attribute를 가짐.

**- E-R Diagrams**

****

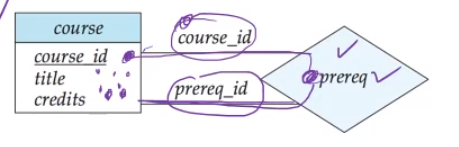
직사각형은 entity set, 다이아몬드는 relation ship을 나타냄.

Attribute들이 entity 직사각형 안에 나열되어 있음.

밑줄은 primary key attribute를 나타냄.

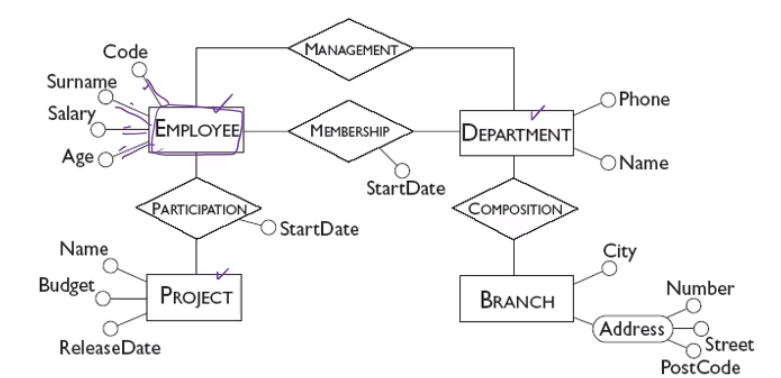
아래와 같이 relationship이 role을 가질 수도 있음. (선수강과목). Unary.

하나의 entity set과 relation을 가지는 관계. 따라서 role을 지정해 줌.



**- An ER Schema**

ER schema의 또다른 형태. 여기선 attribute가 entity에 동그랗게 달려 있음.



**- Types of Attributes**

Simple attribute: value들이 항상 atomic. (relational model)

Composite attribute: 여러 개의 구성요소를 가지는 것.

Ex) name = (lastname, firstname)

Conceptual Design 단계에서는 요구 사항을 정확히 표현하는 게 최우선 과제. 그래서 Relational에서 composite attribute를 허용하지 않더라도 사용할 수 있음.

Null: “Missing or Unknown”. 몇몇 attribute는 null value를 허용하지 않을 수 있음.

Single-valued attribute: 각각의 attribute가 entity에 대한 값을 하나 가지고 있음.

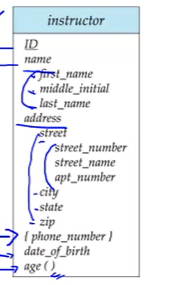
Multivalued attribute: attribute가 인스턴스에 대해 하나 이상의 value를 가지고 있음.

Ex) Children = {john, tom}

Relational model에서 허용 X.

Derived attribute: 다른 attribute혹은 entity의 값으로 해당 attribute 값을 계산.

Ex) duration, count, sum

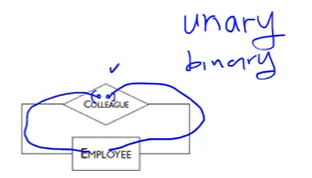
<- { }는 multivalued attribute. ()는 derived. Indent가 들어가 있는 건 composite attribute.

**Degree of a Relationship Set**

대부분의 relationship은 binary임.



Unary, ternary (3개) 등 binary가 아닌 relationship도 정의할 수 있음.

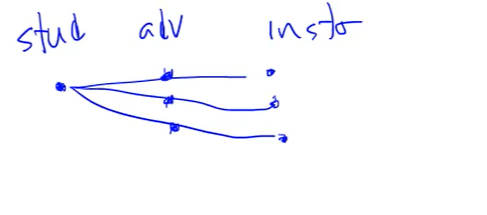
 <- 이건 엄격하게 binary. 결국 2개의 entity를 참조하니까. Binary의 특수한 경우. 근데 unary라고 말하기도 함.

**Mapping Constraints**

Degree: relationship의 schema 적인 모습. (몇 개의 attribute set과 연결이 되어있는가)

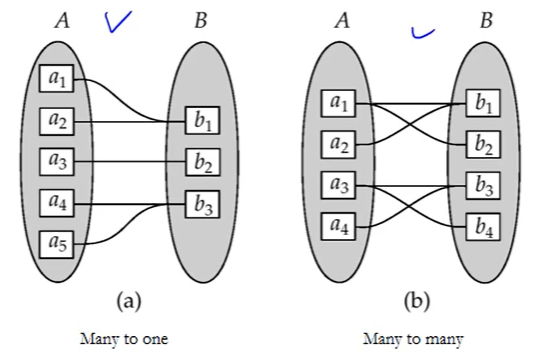
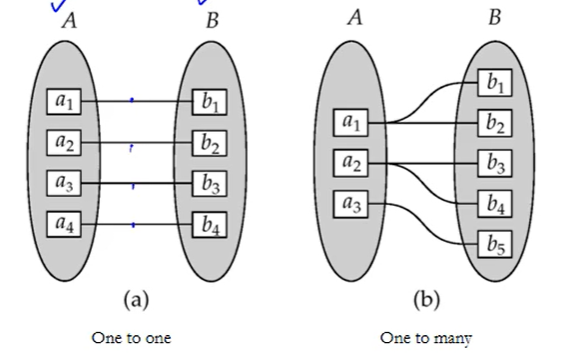
Cardinality: entity가 relationship set을 통하여 다른 entity set과 몇 개나 연결이 되는지.

**그러니까 degree는 relation에 있는 attribute 개수. Cardinality는 연결된 entity의 개수.**

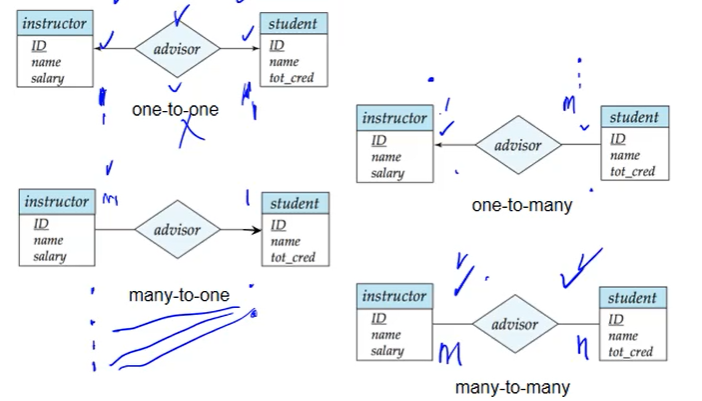


1:1, 1:m, m:1, m:n 이 있음.

**Mapping Cardinalities**



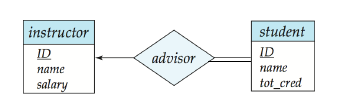
**Mapping (Cardinality) Constraints**

 <- 화살표가 1을 의미. 이 때 1이나 m은 max를 의미. 즉 무조건 관계가 있어야 하는 건 아님. 모든 record가 관계가 있는 건 아니고 관계가 있다면 저런 관계라는 뜻.

**Participation of an Entity Set in a Relationship Set**

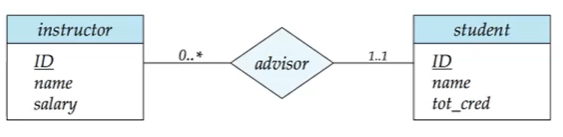
Total participation (두 줄): min 값을 제한하는 것. Entity set의 모든 entity가 최소 하나의 relationship을 가져야 한다.

Partial participation (한 줄): 몇몇 entity는 관계에 참여하지 않을 수도 있음.



모든 학생은 지도교수가 있어야 하고, 또 여러 명의 학생이 한 명의 지도 교수를 가질 수 있음. 근데 교수님들은 지도 학생이 없는 경우가 있을 수도 있음.

**Alternative Notation for Cardinality & Participation**

****

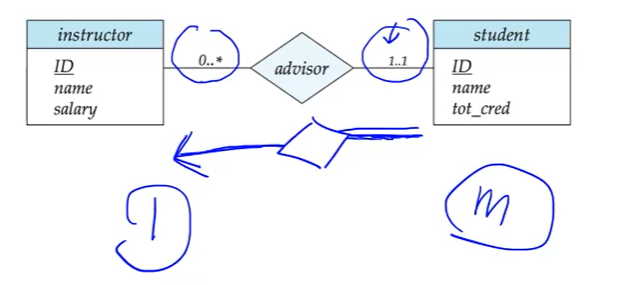
위의 그림의 다른 버전.

Instructor가 advisor relationship에 최소 0번, 최대 m번 참가.

Student는 최소 1번, 최대 1번 참가.

1:M relationship. 왜냐면 instructor가 여러 번 참가한다는 의미는 하나의 instructor가 여러 명의 student를 가질 수 있다는 의미. 따라서 student 쪽이 many임.

따라서 화살표 표기법으로 바꾸면 instructor 쪽에 화살표가 감.



**Keys**

- Key for an Entity (Set)

relational model의 key와 같음: super key, candidate key, primary key.

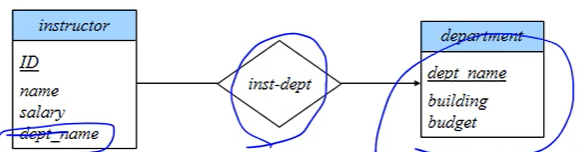
다른 entity와 구분할 수 있는 attribute의 집합

- Key for a Relationship (Set)

연관된 Entity set의 Primary key의 combination이 super key를 이룸. Minimal 하다면 candidate. Candidate key(혹은 primary key)를 정할 땐, **mapping cardinality**를 고려해야 함.

**Redundant Attributes**

Entity의 attribute가 relationship에 있을 경우, 제거되어야 함. Redundancy는 inconsistency를 야기하기 때문.

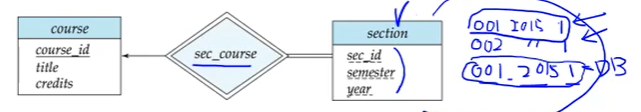


NOTE: dept\_name은 Foreign key가 아니냐? 하지만 relational model은 관계를 명시적으로 표현할 방법이 없기 때문에 간접적으로 표현하기 위해 foreign key로 넣은 것이고, ER모델은 relation을 표현 가능하기 때문에 foreign key 역할을 하는 attribute를 따로 둘 필요가 없음.

**Weak Entity Sets**

Weak entity set: primary key를 형성하기에 충분한 attribute를 가지고 있지 않은 entity set.

<-> Strong entity set.

 <- section이 weak entity set. 따라서 entity를 구분하려면 course와의 relationship에 의존.

Weak entity set의 존재는 identifying entity set의 존재에 의존한다.

- Identifying relationship은 **더블 다이아몬드**로 나타냄.

Weak entity set의 Discriminator (혹은 partial key)는 같은 strong entity에 연관된 다른 weak entity들을 구분할 수 있는 attribute의 집합.

**Primary-key(weak entity set) = primary\_key(identifying strong entity) U discriminator(weak entity set)**

**Extended E-R Features**

- Specialization

Entity set 안에서 subgrouping을 하는 것.

공통의 attribute를 공유함.

각각 sub entity는 고유한 attribute를 가질 수도 있음.

- Generalization

같은 기능을 공유하는 여러 Entity set를 묶어서 high level의 entity set을 만드는 것.

둘은 방향이 반대인 같은 기능이라고 생각하면 됨. 어디서 시작하는지에 따라 다름.

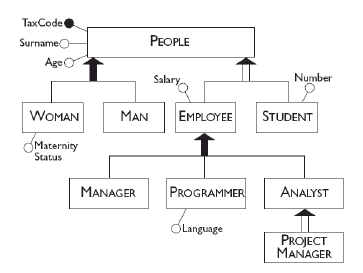
하위 entity는 상위 entity의 attribute를 모두 가짐.

Inheritance: high-level entity set의 attribute와 relationship은 lower-level entity set으로 상속됨.

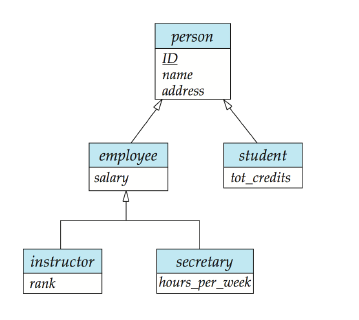
Types of Generalization (super-sub entities)

Disjoint vs overlapping: entity가 두 개 이상의 sub entity set에 포함될 수 있는지. 즉 sub entity끼리 겹치면 overlapping, 서로 독립이면 disjoint.

Total vs partial: 모든 higher level entity가 lower level entity set에 속하는지. 즉 상위 entity가 sub entity들로만 이루어지면 total, 아니면 partial.



위의 그림에서 검정색 굵은 화살표가 total이고 속이 빈 화살표가 partial인듯.



위 그림에서 화살표가 뭉친게 total이고 따로인게 partial?

애매하면 관계에 그냥 total, disjoint 같은 걸 글로 적으면 됨.

**Reducing ER schema to tables**

Logical design: conceptional design(ER 모델)을 relational model로 mapping.

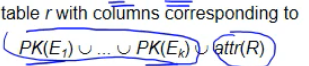
기본 룰:

각각의 entity set: 고유의 테이블

각각의 relationship set: 고유의 테이블

Relationship set R

E1부터 Ek를 relate하면,

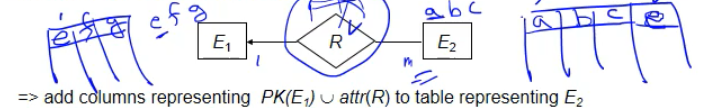


각 entity의 primary key를 합하고, relation의 attribute를 더함.

Special case: 1:m과 1:1.

1:m이나 1:1일 때 굳이 relation을 만들지 않아도 됨.

즉 관계에서 m:1, 1:1 등 1이 있으면 foreign key로 나타낼 수 있음.



Many 쪽에 E1의 primary key를 합해서 테이블을 만들면 됨.

1쪽에 하면 하나에 여러 개의 many가 붙으므로 ㄴㄴ

**Generalization/Specialization to Relation Schema**

옵션 1: 각 entity set마다 하나의 테이블을 만듦.

Relationship은 foreign key로 번역됨. 왜냐? 상위 entity가 존재하기 때문에 이 entity를 참조키로 넣어줘야 함.

Partial 이거나 overlapping인 generalization에 좋음. - Partial이면 하위 entity로 상위 entity를 모두 나타내지 못하므로, overlapping이면 데이터의 중복 때문에.

옵션 2: 낮은 레벨의 entity set만 보관함.

더 높은 entity set의 attribute를 lower level의 entity set의 attribute에 합침.

Total 이거나 disjoint인 generalization에 좋음. – 위의 이유와 반대.

**Design Issues**

Entity vs Attribute

만약 employee의 telephone이 있을 때, Attribute로 나타내면 simple, entity로 나타내면 independent.

결정은 telephone이 독립적인 entity로 취급되어야 하는지, employee가 가질 수 있는 telephone의 개수, telephone이 employee 사이에 공유되는지 등에 따라 이루어져야 함.

Entity vs Relationship

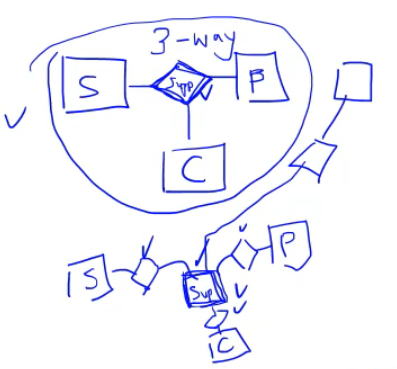
“customer having an account at a branch”

Account를 relationship으로 나타내면 간단하지만 제한되어 있음. (다른 relationship에 참여할 수 없다.). **즉 둘 사이의 관계 외에 다른 기능을 가질 수 없음.**

Account를 entity로 나타내면 account는 분리된 entity로 행동할 수 있음.

Binary vs n-ary relationships

모든 n-ary relationship은 binary relationship으로 나타낼 수 있음: entity와 그 entity 연관된 relationship을 추가해서.



하지만 항상 바람직한 건 아님.

Generalization and Specialization

따로따로 만들 지 generalization을 할 지… 정답은 없음.

결정은 모델이 어떻게 실제 세계를 표현하는 지에 따라 이루어져야 함!

Correct, Complete, Readability.