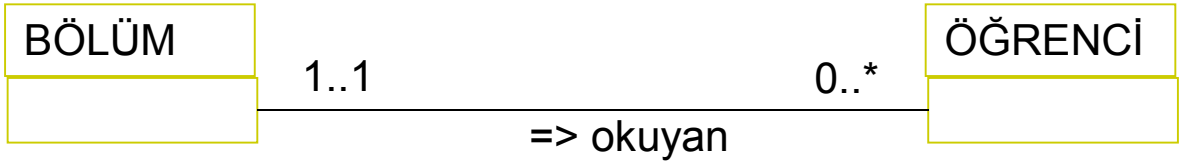
- **Bir şirkette sipariş-işleme (order-processing) VT uygulaması:**
- Şirketin çok sayıda müşterileri (customer) var. Müşteri nitelikleri customer id, customer name ve city. Her müşteri diğerlerinden customer id ile ayırt ediliyor.
- Şirket müşterilerine sipariş doğrultusunda ürünler (items) satıyor. Her ürün için, biricik(unique) item no, item name, birim fiyat (unit price) bilgisi tutuluyor.
- Şirketin birçok deposu (warehouses) var. Depo ile ilgili nitelikler biricik (unique) warehouse id ve bulunduğu şehir.
- Müşteriler birçok üründen oluşan bir sipariş (order of many items) verdiğinde buna ilişkin yeni bir sipariş no üretiliyor ve tarih ile birlikte tutuluyor. Her üründen istenilen sayıda sipariş edilebiliyor.
- Siparişin alınması üzerine, şirket bu sipariş edilen ürünleri temin ederek müşteriye gönderiyor (shipment yada delivery). Gönderilme tarihi (ship date) bilgisi tutuluyor.
- a) Bu gereksinim analizi çerçevesinde istenilen veritabanının varlık-bağıntı (entity-relationship) diyagramını çiziniz. Birincil anahtarlar (primary keys) ve bağıntı kısıtlamalarını belirtiniz.

Participation Constarint

- Does every department have a manager?
  - this is a participation constraint: the participation of Departments in Manages is said to be total (vs. partial) (tam katılım yada kısmi katılım)
  - Every did value in Departments table must appear in a row of the Manages table (with a non-null ssn value!)
- *B varlık kümesindeki her bir ‘b’ varlığı, A varlık kümesindeki bir ‘a’ varlığı ile mutlaka bağıntılı olması gerekiyorsa, «B varlık kümesi A varlık kümesine var olma bağımlıdır» denir. Burada A birincil varlık, B ikincil varlık olarak adlandırılır.*
- *Bölüm, Çalışan’a var olma bagımlıdır.*

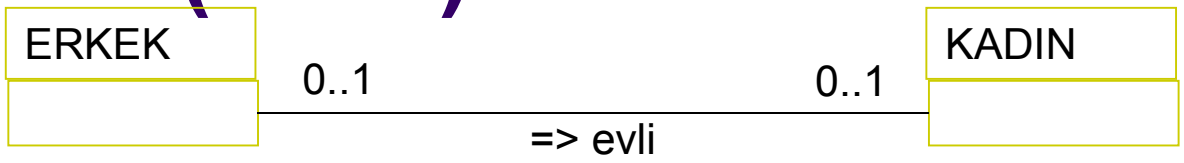
Örnek

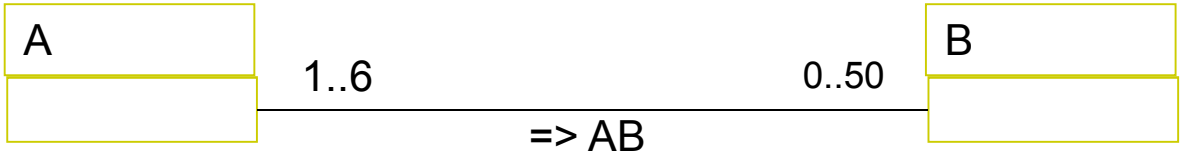


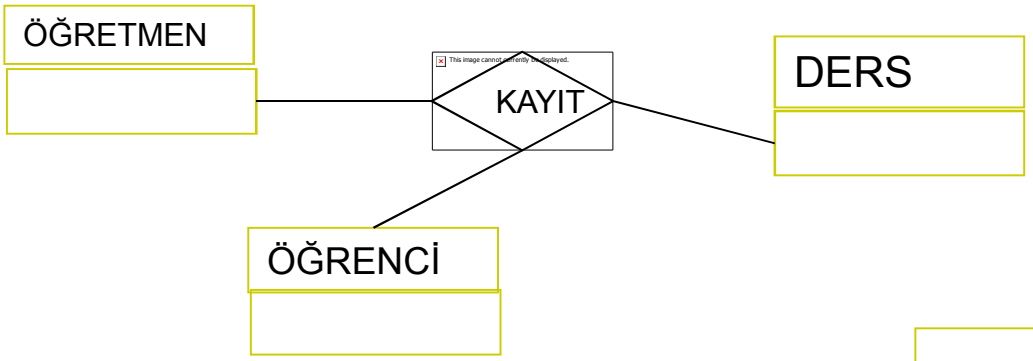
Weak Entities

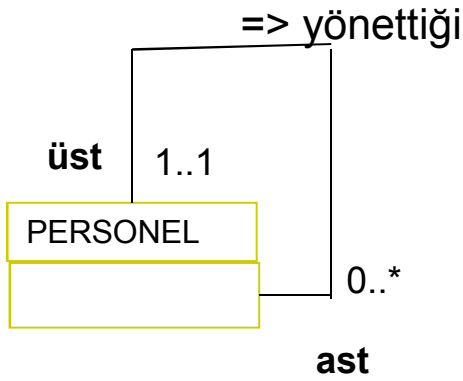
- A weak entity can be identified uniquely only by considering the primary key of another (owner) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this identifying relationship set.

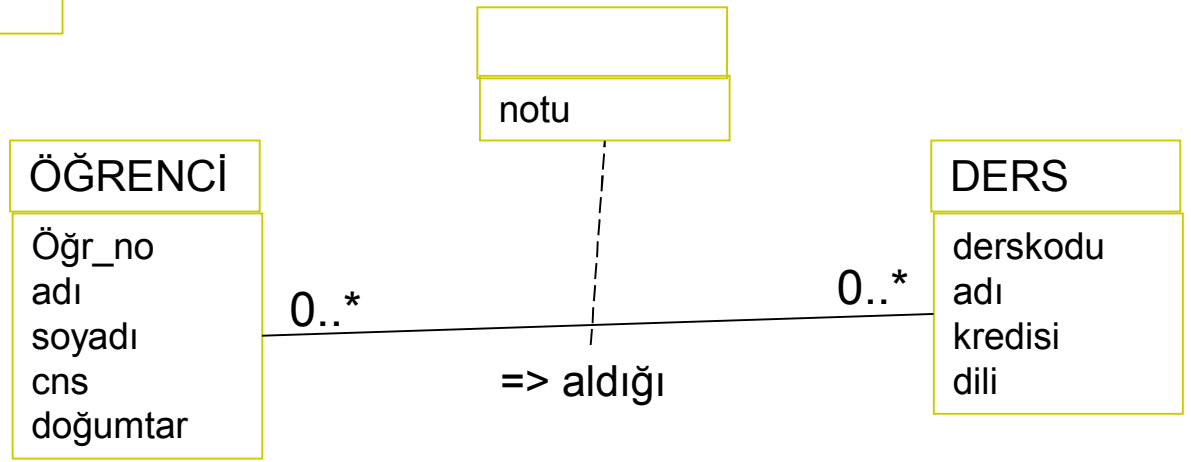
Örnek (UML)


ERKEK
KADIN
0..1
0..1
=> evli


A
B
1..6
0..50
=> AB

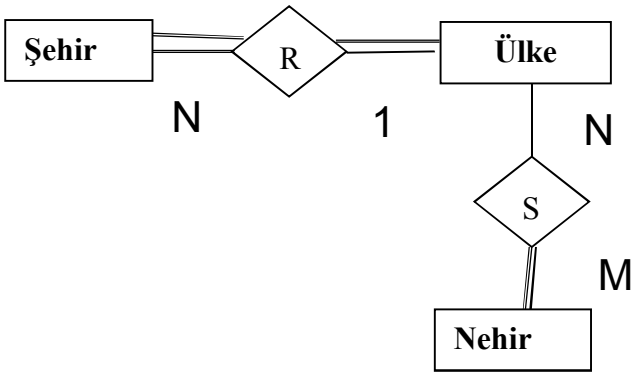

ÖĞRETMEN
KAYIT
DERS
ÖĞRENCİ

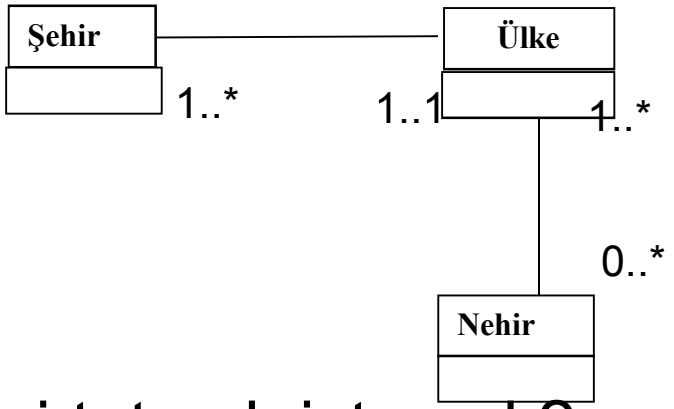

=> yönettiği
üst
1..1
PERSONEL
0..*
ast


notu
ÖĞRENCİ
Öğr_no
adı
soyadı
cns
doğumtar
0..*
0..*
=> aldığı
DERS
derskodu
adı
kredisi
dili

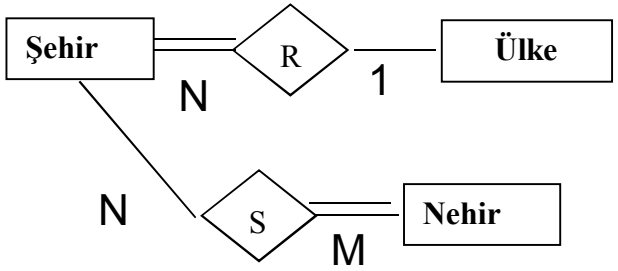
Örnek (ER,UML)

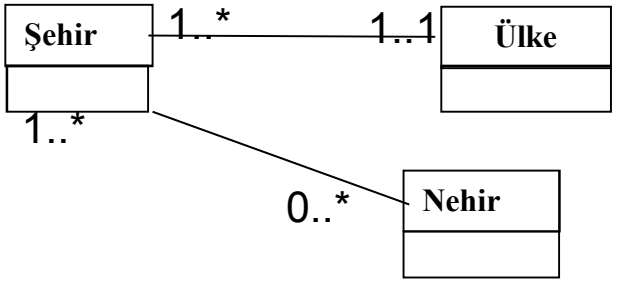
- Şehirler, ülkeler ve nehirleri gösteren bir veritabanı tasarımı.





- Nehirlerin hangi şehirlerden geçtiğini tutmak istersek?





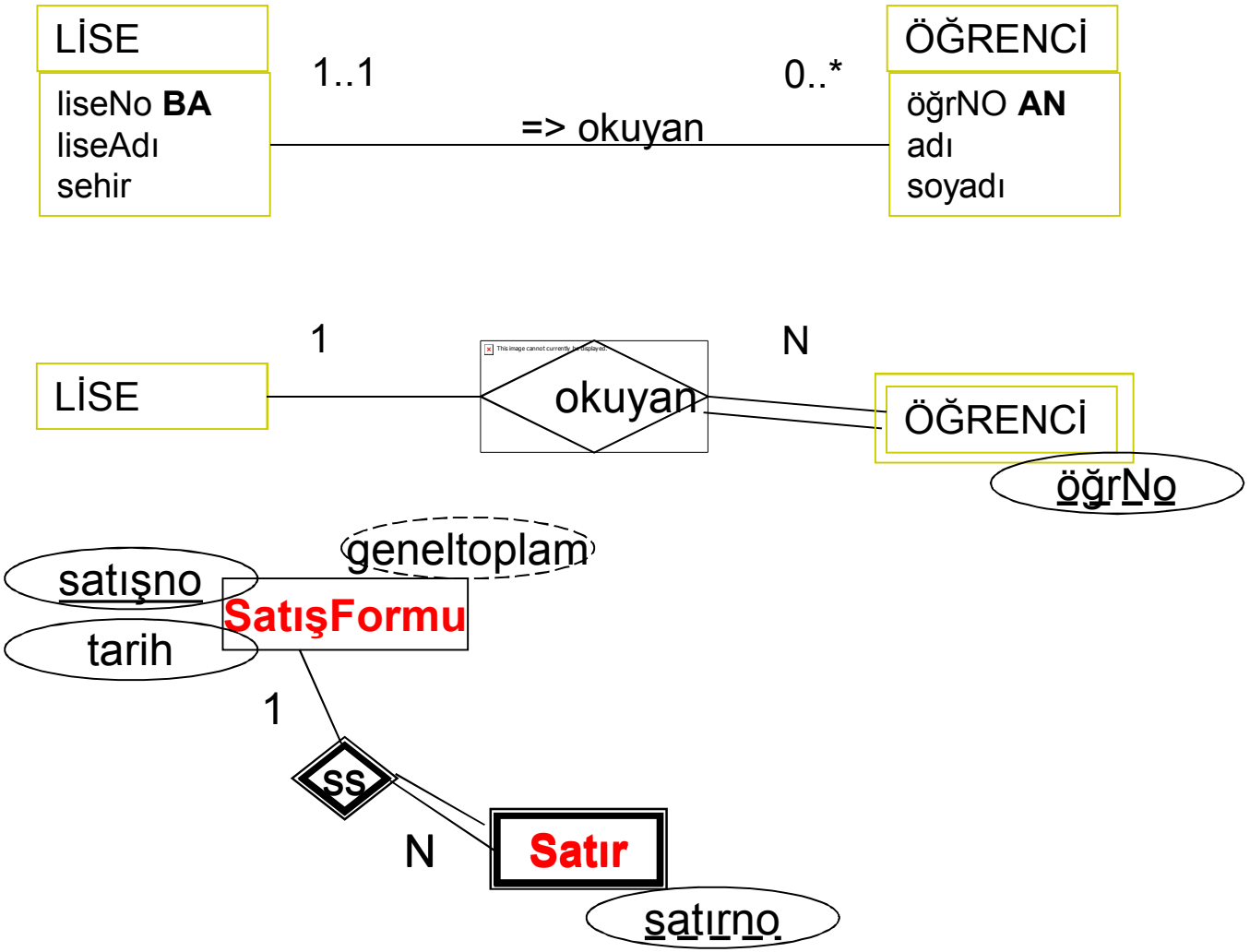
Güçlü/Zayif Varlık Kümeleri

- Bir varlık kümesinde anahtar bulunamıyorsa *(bütün nitelikler birarada olsa da anahtar olmuyorsa)* bu varlık kümesine **«zayıf varlık kümesi»** denir.
- Anahtarı olan varlık kümesine «güçlü varlık kümesi» denilir.

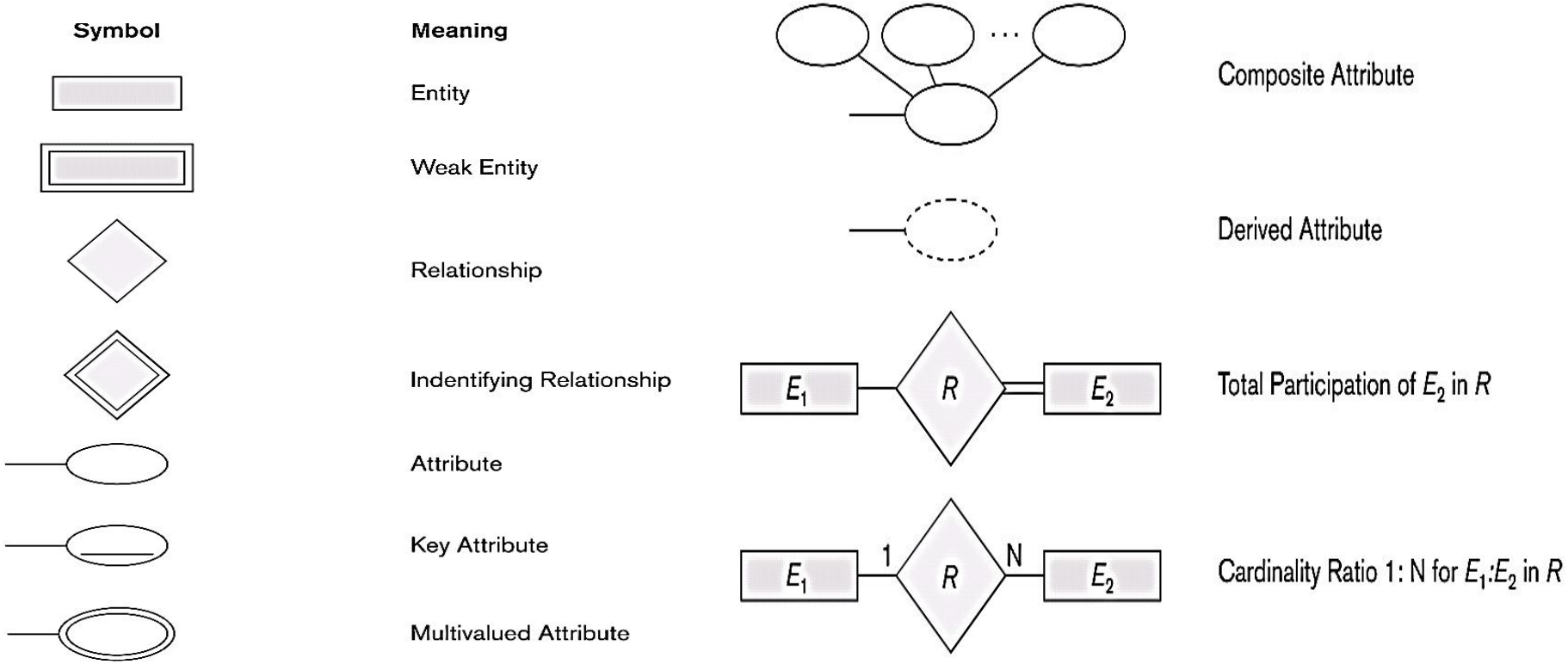
- Zayıf varlık kümesi, güçlü bir varlık kümesi ile 1-1 veya N-1 tipinde var olma bağımlı bir bağıntı kurmalıdır.

- Zayıf varlık kümesindeki, aynı güçlü varlığa bağlı olan varlıkları ayırd edebilmek için ayırdedici nitelik(ler)-**AN** tanımlamak gerekir.

zayıf varlık kümesi örnekleri



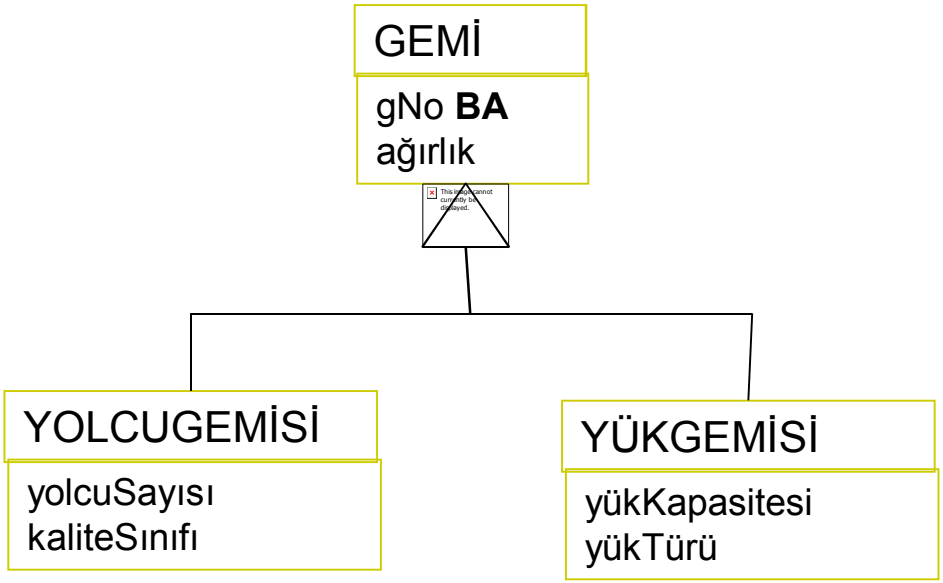
Chen notasyonundaki sembol ve anlamları



ISA (`is a') Hierarchies

- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.
  - Overlap constraints(Ayrıklık kısıtlaması):Can Joe be an Hourly_Emps as well as a Contract_Emps entity?(Allowed/disallowed)
  - Covering constraints(Katılım kısıtlaması): Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity?(yes:mandatory/no:optional)
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify entities that participate in a relationship.

Genelleme *(«IS A», «Ait olma» bağıntısı)*



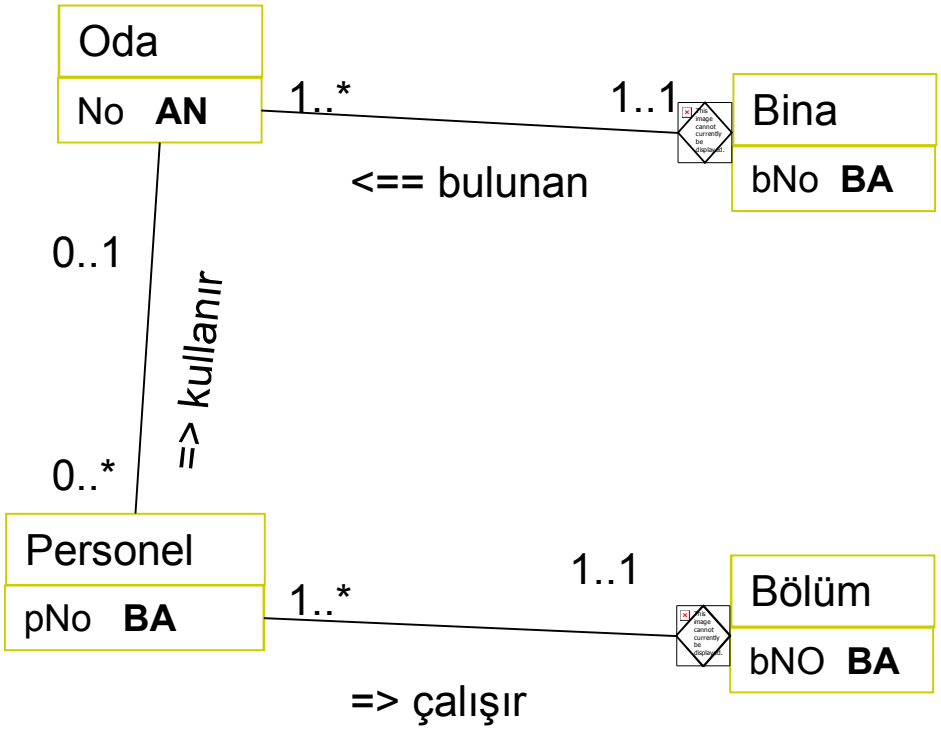
- Kalıtım: nitelik üst varlıktan alt varlığa geçer
- Katılım kısıtlaması:
  - Zorunlu: üst sınıfın her ögesi alt sınıfta yer alır
  - Seçimli: yer almayabilir
- Ayrıklık kısıtlaması:
  - OR: üst sınıfın her ögesi en çok bir alt sınıfta var (alt sınıflar: disjoint-ayrık)
  - AND: birden çok alt sınıf
- 4 farklı kısıtlama tanımlanabilir, ihtiyaca göre kullanılabilir.

Aggregation

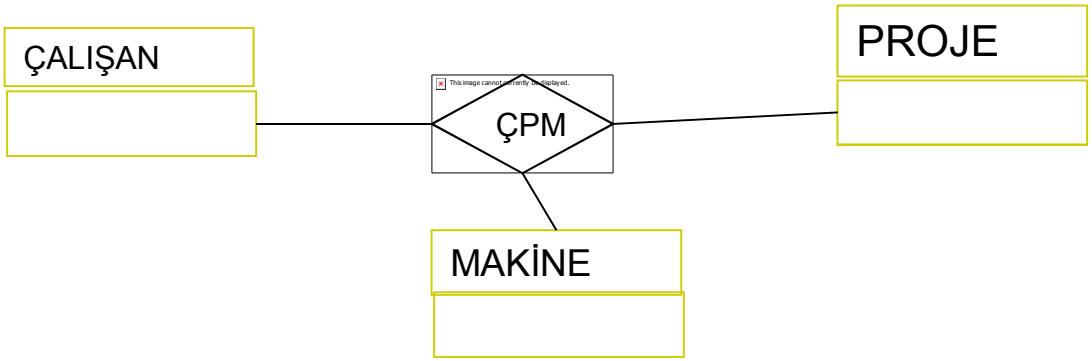
- Used when we have to model a relationship involving (entitity sets and) a relationship set.
- Aggregation allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.

Kümeleme-1 (*aggregation*)

- «has a», «bulunur/sahiptir» bağıntısı
- ‘Küme’ ile ‘Parçaları’ arasındaki bağıntılar için kullanılır. Parçalar kümeye **var olma bağımlıdır**.



Kümeleme (çok dereceli bağıntılar)



- Yukarıdaki 3'lü bağıntıda:
  - İki varlık arasındaki bağıntı, 3. varlıktan bağımsız bir şekilde ifade edilemiyor.
  - (Kavramsal seviyede de olsak) Bilgi tekrarı var.
  - Bilgi tekrarından dolayı aykırılık/anormallikler ortaya çıka(bili)r.
- **Kümeleme**, varlık kümeleri ile aralarındaki bağıntının kümelenmesi ile yeni bir varlık kümesi tanımlamaya olanak sağlar.

Conceptual Design Using the ER Model

- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.

Entity vs. Attribute

- Should address be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends on the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, address must be an entity (since attributes cannot be set valued).
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, address must be modeled as an entity (since attribute values are atomic).

Summary of Conceptual Design

- Conceptual design follows requirements analysis,
- Yields a high-level description of data to be stored
  - ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
  - Basic constructs: entities, relationships, and attributes (of entities and relationships).
- Some additional constructs: weak entities, ISA hierarchies, and aggregation.
- Note: There are many variations on ER model.

Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: key constraints, participation constraints. Some foreign key constraints are also implicit in the definition of a relationship set.
- Some constraints (notably, functional dependencies) cannot be expressed in the ER model.
- Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Contd.)

- ER design is subjective. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise.

  Common choices include:

  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and whether or not to use aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.

Örnek: COMPANY Database

- We need to design a database schema based on the following (simplified) **requirements** of the COMPANY Database:
  - The company is organized into DEPARTMENTs. Each department has a name, number and an employee **who *manages* the department**. We keep track of the start date of the department manager. A department may have several locations.
  - Each department ***controls*** a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.
  - We store each EMPLOYEE's social security number, address, salary, and birthdate.
    - Each employee ***works for*** one department but may ***work on*** several projects.
    - We keep track of the number of hours per week that an employee currently works on each project.
    - We also keep track of the ***direct supervisor*** of each employee.
  - Each employee may ***have*** a number of DEPENDENTs.
    - For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

Initial Design of COMPANY DB

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - DEPARTMENT, PROJECT, EMPLOYEE and DEPENDENT
- The initial attributes shown are derived from the requirements description

DEPARTMENT
number **BA** name manager managerStartDate locations

PROJECT
number **BA** name location controllingDept

EMPLOYEE
Ssn **BA** fName lName department birthDate salary worksOnProjects supervisor adress

DEPENDENT
name dependsToEmp birthDate relationships

İlk tasarım sonrası iyileştirmeler

- *Eğer mevcut nitelik başka bir varlığa işaret ediyorsa:*
  - Nitelik➔ Bağıntı haline dönüşmeli.
- *Eğer mevcut nitelik çok değer alabiliyorsa:*
  - Nitelik➔ Varlık kümesi haline dönüsmeli.
- *Eğer mevcut varlık kümesi sadece 1 niteliğe sahip ve bir bağıntısı varsa:*
  - Varlık kümesi ➔ nitelik haline dönüşmeli.
- Varlık kümeleri arasında «genelleme» mümkünse yapılmalı. Yukardan-aşağıya (top-down), aşağıdan-yukarıya (bottom-up) yaklaşımlar kullanılmalı.
- Bağıntıların dereceleri tekrar degerlendirilmeli, gerekirse «kümeleme» ile daha açık yapılanmalar kullanılmalı.

COMPANY DB (devam)

- In the refined design, some attributes from the initial entity types are refined into relationships:
  - Manager of DEPARTMENT -> MANAGES
  - Works_on of EMPLOYEE -> WORKS_ON
  - Department of EMPLOYEE -> WORKS_FOR
  - Controlling_Department of PROJECT → CONTROLS
  - Supervisor of EMPLOYEE → SUPERVISION
  - Dependent_name of DEPENDENT → DEPENDENTS_OF

- In general, more than one relationship type can exist between the same participating entity types(MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT (Different meanings and different relationship instances.)

COMPANY DB ER DIAGRAM

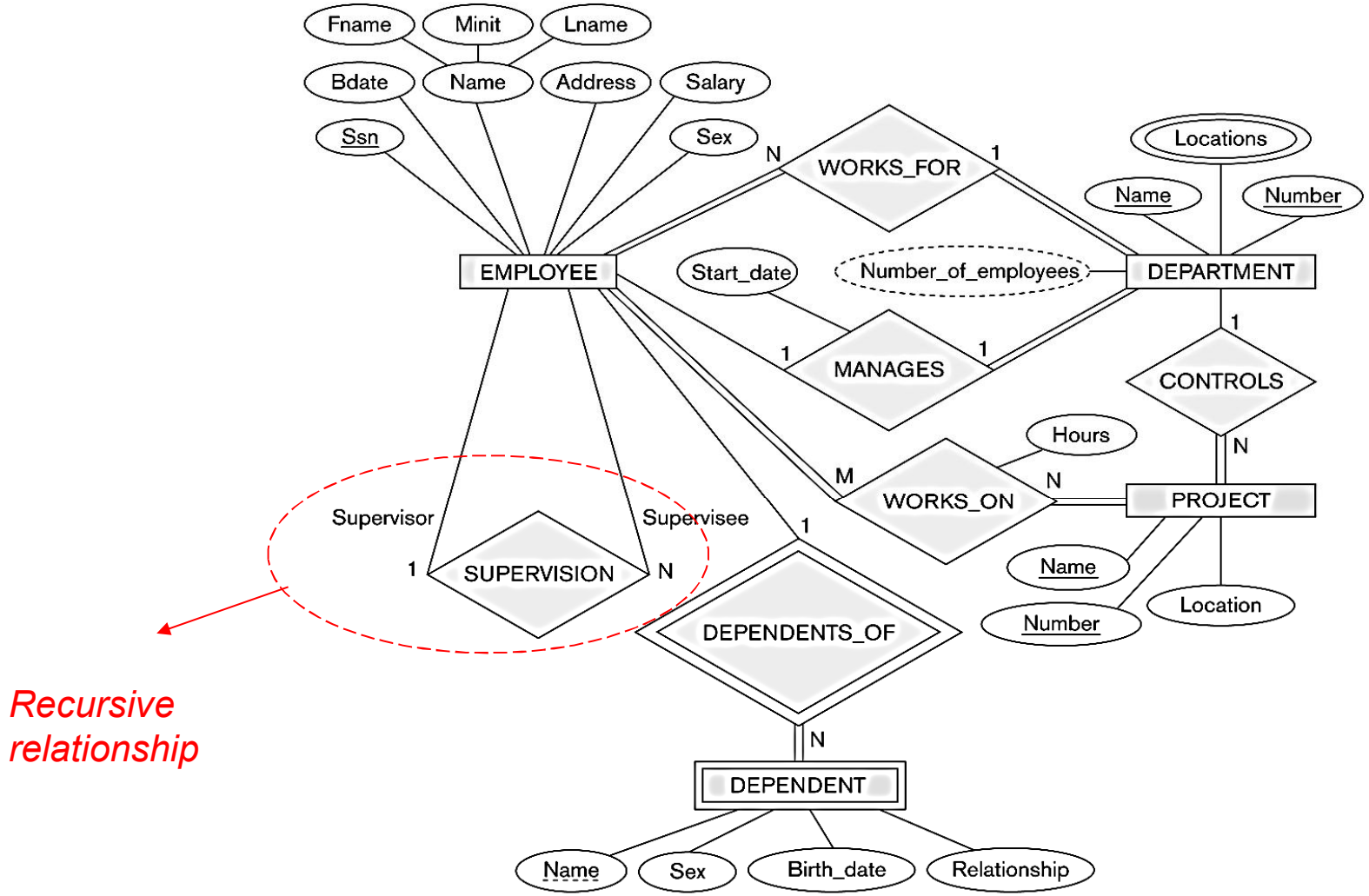


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.