Ders # 2

VERİ TABANI DERS NOTLARI

Veri Tabanı Tasarımı

Varlık Bagıntı ile veri modelleme (data modeling with EER, UML)

- Ünal Yarımağan, "Veri Tabanı Sistemleri", Akademi Yayınevi, 2010
- Elmasri, Navathe, "Fundamentals of Database Systems", 5th Edition, Addison Wesley, 2008
- Toby Teorey, Sam Lightstone, Tom Nadeau, H.V. Jagadish,"DB Modelling & Design, Logical Design", 5th Edition, Morgan Kaufmann, 2011

Genel Bakış

- Veri tabanı tasarım prosesi genel şeması
- Temel Kavramlar
  - Varlık, Varlık kümesi ve nitelikleri, Bağıntı, Bağıntı kümesi, bağıntı sınırlamaları ve bağıntı türleri, Rol tanımlama
  - Anahtar nitelik
  - Zayıf Varlık Kümeler
- Yardımcı Kavramlar
  - Genelleme
  - Kümeleme, Kompozisyon
- Örnek bir VT :(COMPANY veri tabanı)
- ER ve UML veri modeli ile örnekler

VT Tasarım Prosesi

- VT tasarımı
  - Veri modelleme
  - Veri mühendisliği, bilişim uzmanlığı ile ilgili
- Uygulama tasarımı
  - Program akış ve arayüzlerin tasarımı
  - Yazılım mühendisliği ile ilgili

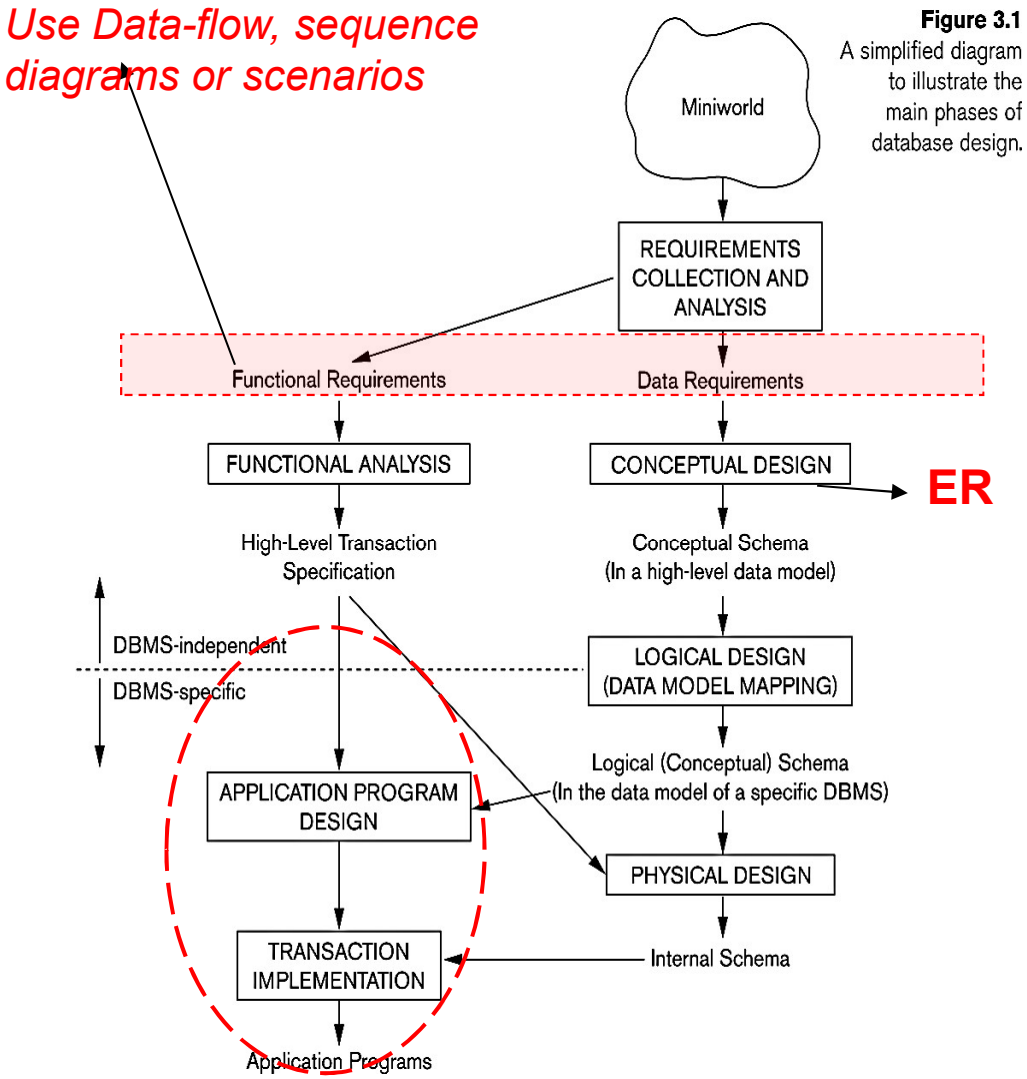
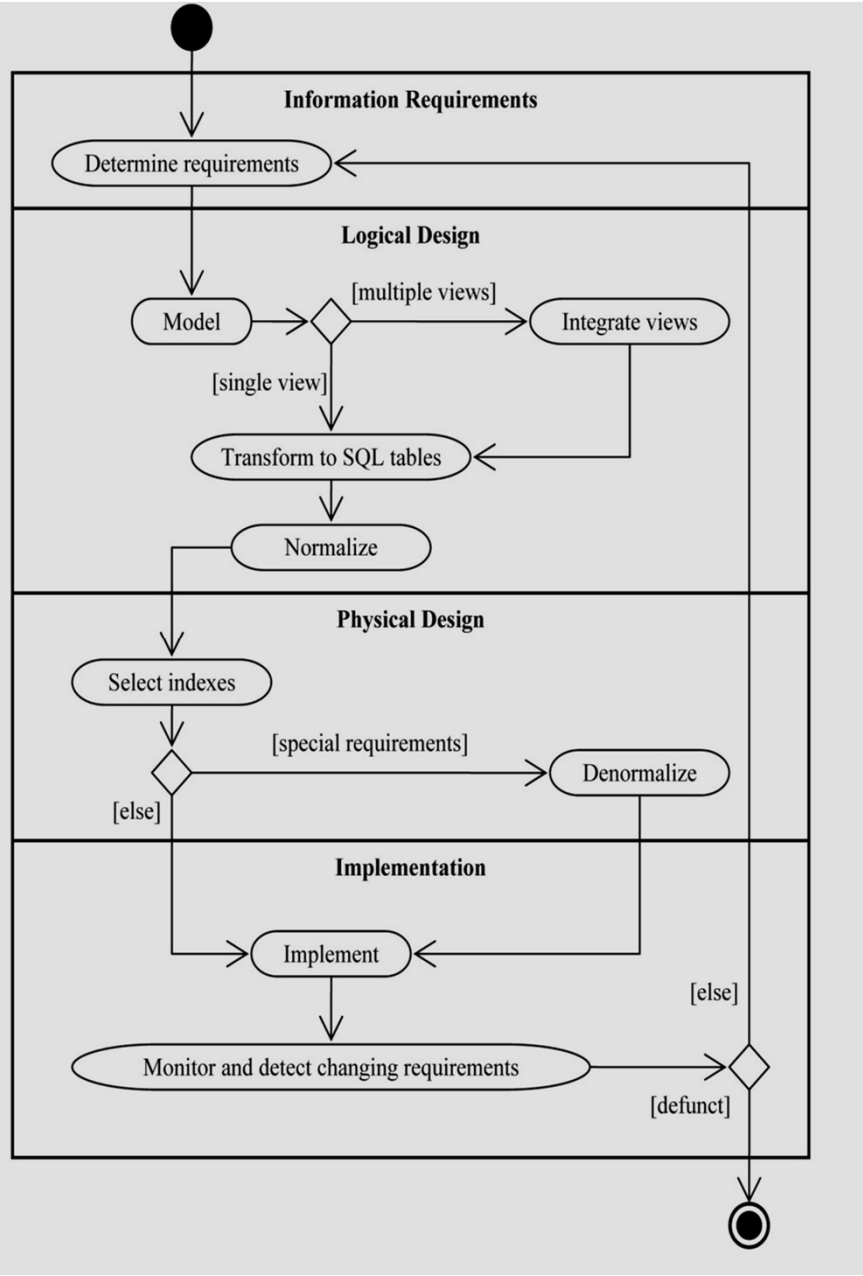
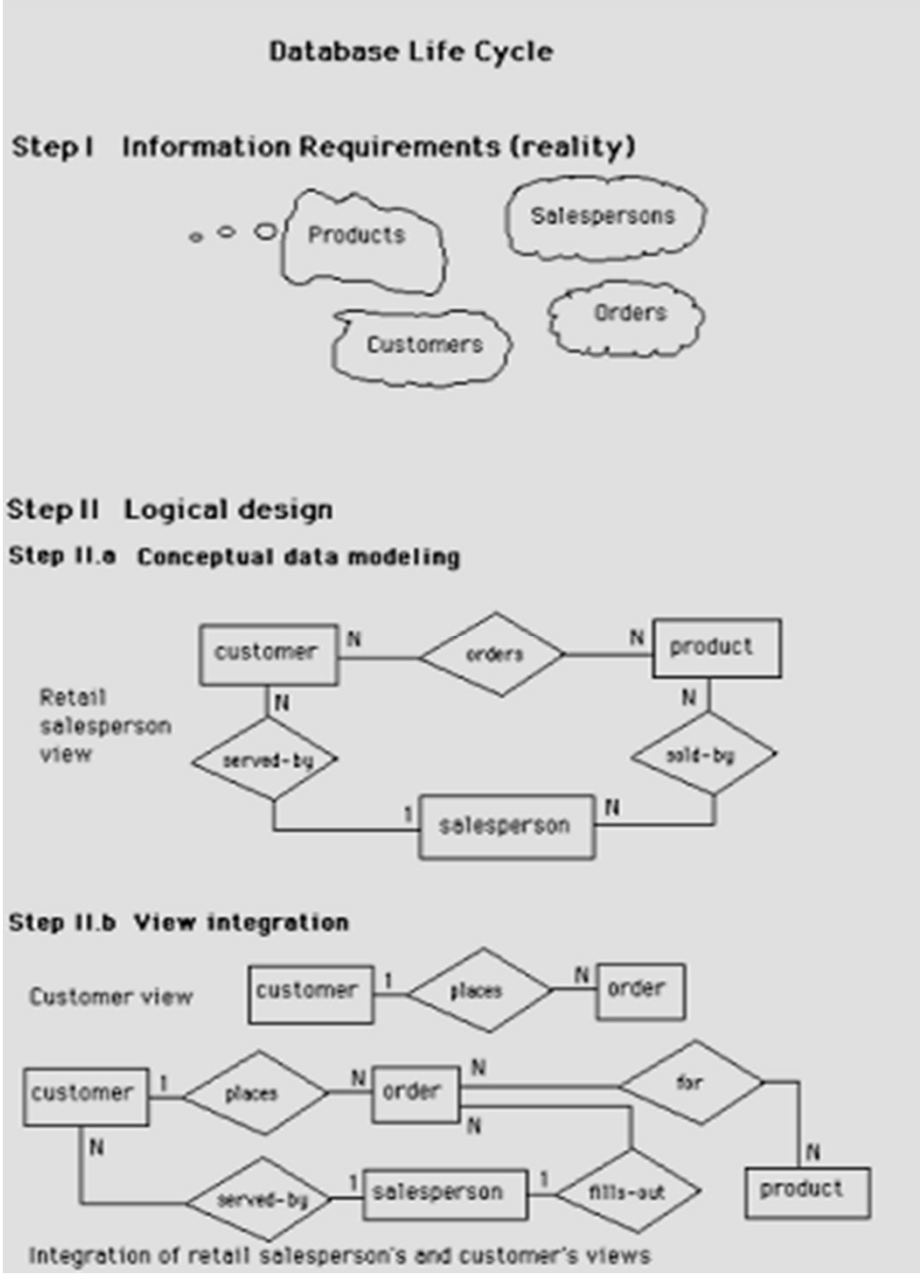


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

VT Tasarım Prosesi





VT Tasarımı

- **Gereksinimlerin toplanması ve analizi (Requirements collection and analysis)**
  - VT Tasarımcısı, veri gereksinimlerini anlamak ve belgelemek için müstakbel veri tabanı kullanıcıları ile görüşmeler (interview) yapar
  - Çıktı: **veri gereksinimleri**
  - Uygulama için işlevsel gereksinimler **(Functional requirements)**

VT Tasarımı

- **Kavramsal şema**
  - Kavramsal bir tasarım
  - Veri gereksinimlerini açıklamalı
  - Varlık türleri detaylı açıklama (entity types)
  - Bağıntıları detaylı açıklama (relationships)
  - Kısıtlamaları ayrıntılı açıklama (constraints)
  - Üst-düzeyli veri modelinden gerçekleştirim veri modeline dönüşüm (Transformed from high-level data model into implementation data model)

VT Tasarımı

- **Mantıksal tasarım veya veri modeli eşlemesi (data model mapping)**
  - Result is a database schema in implementation data model of DBMS
- **Physical design phase**
  - Internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files specified

Varlık Bağıntı (Entity-Relationship, ER) Modeli

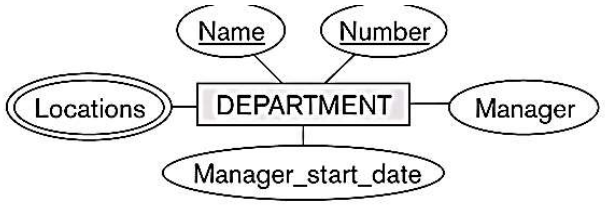
- ER modelde verileri tanımlama:
  - Varlıklar (Entities)
  - Bağıntılar (Relationships)
  - Nitelikler (Attributes)

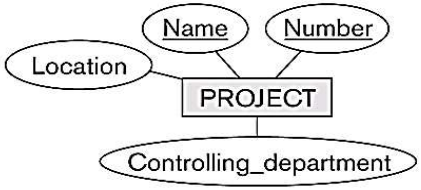
- UML diyagramları (*UML* notasyonu)
  - Yazlım geliştirme metodolojisinde kullanıır.
  - ER'daki varlık UML nesnesine karşılık gelir.
  - UML nesnesinde 3 kısım var: isim, nitelikler ve operasyonlar.

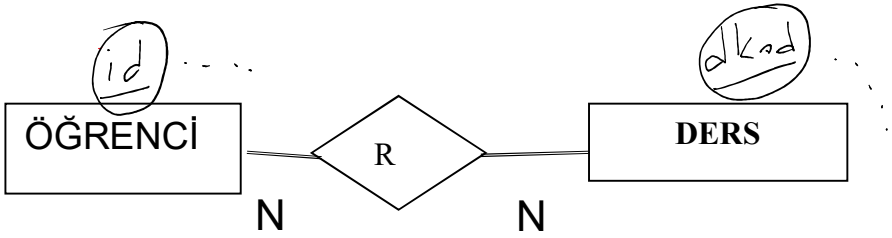
Varlık/Bağıntı kümeleri

- Var olan ve diğerlerinden ayırdedilebilen her nesneye varlık denir. Benzer varlıkların oluşturduğu kümeye varlık kümesi denir.
- Varlıkların nitelikleri vardır. Bu nitelikler ile varlıklar birbirinden ayırdedilir.
  - *EMPLOYEE (İŞÇİ)*
    - *John Smith, Research , ProductX*
  - Her niteliğin bir değer alanı, yani olurlu değerlerinin tümünü içeren bir küme (*domain*) vardır. Niteliğin tipi, değer aralığı, formatı bu kapsamdadır.
- İki ya da daha çok varlığın arasındaki «olayı» tanımlamaya yarayan kavrama bağıntı denir. Aynı türdeki benzer bağıntıların kümesine ise bağıntı kümesi denilir.
  - ÖĞRENCİ DERSİ ALIYOR ise
    DERS ile ÖĞRENCİ arasındaki bağıntı

Temel yapı taşları: varlık, bağıntı







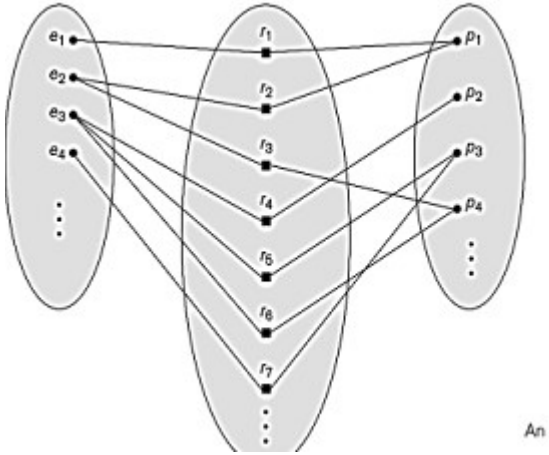


Figure 3.13
An M:N relationship,

Composite attribute

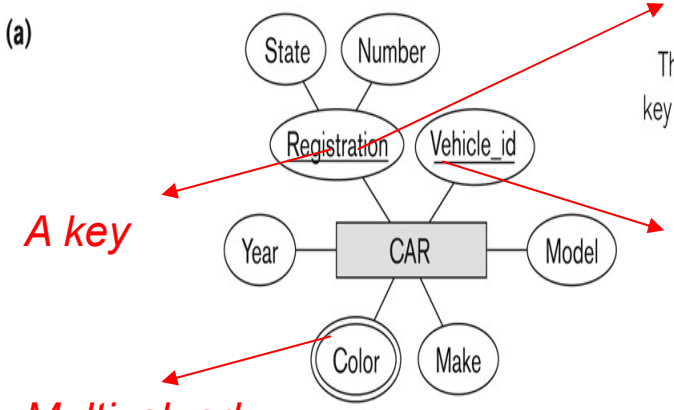


Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

A key

A key

Multivalued attr.

(b)

CAR

Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

three CAR entity instances in the entity set for CAR

Bağıntı Küme sınırlamaları

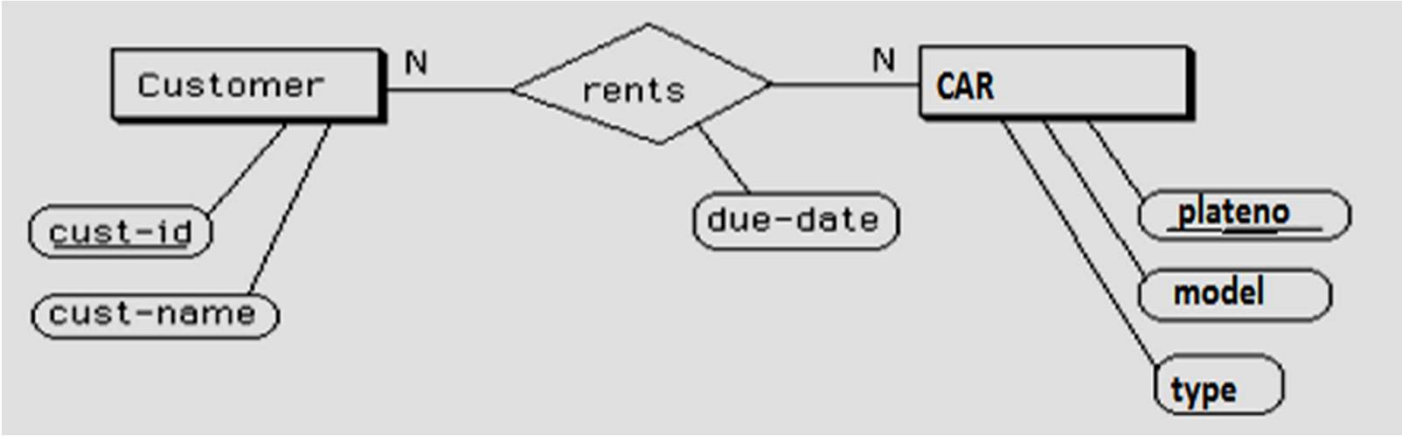
Bağıntı ismi *(relationship name):* *manaya uygun bir isim*

Bağıntıdaki varlıklar *(participating entity types):* *2 veya daha çok, aynı veya farklı varlıklar arasında*

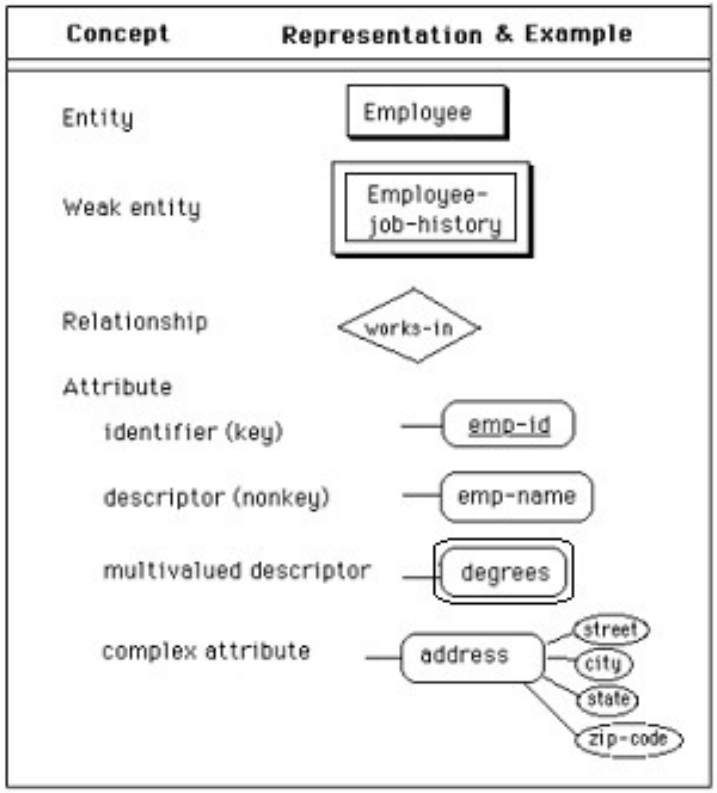
Bağıntının derecesi *(degree of a relationship)*: *2,3,...*

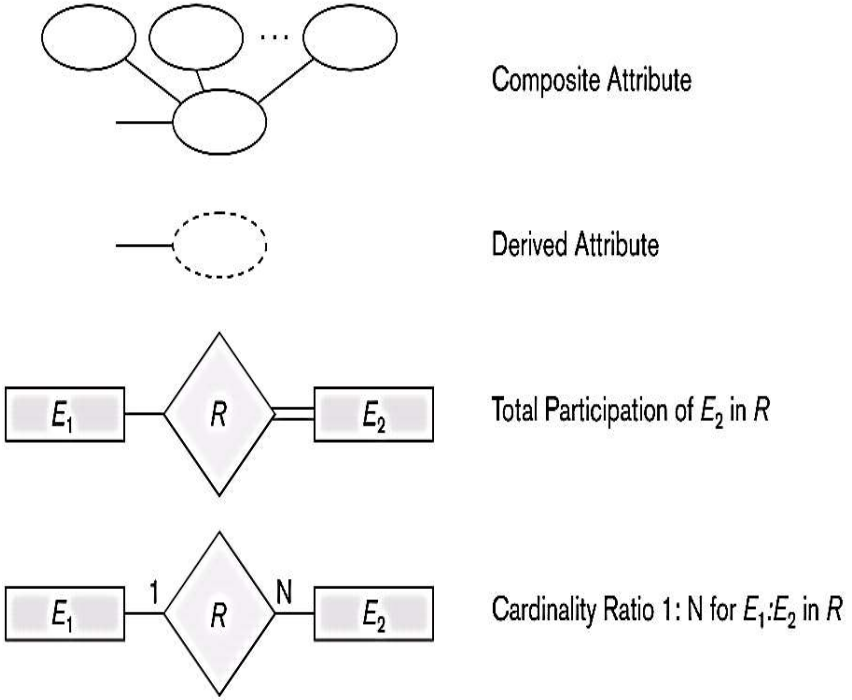
Eleman sayıları,kaça-kaçlık olduğu *(cardinality ratio)*: *1-1, 1-N, N-1, N-N*

ROL tanımlama: *Bağıntıdaki varlıkların bağıntıdaki işlevlerini belirleyen bir rolleri vardır. Bunun açık, anlaşılır olmadığı durumlarda belirtilmesi gerekir.*



ER notasyonları

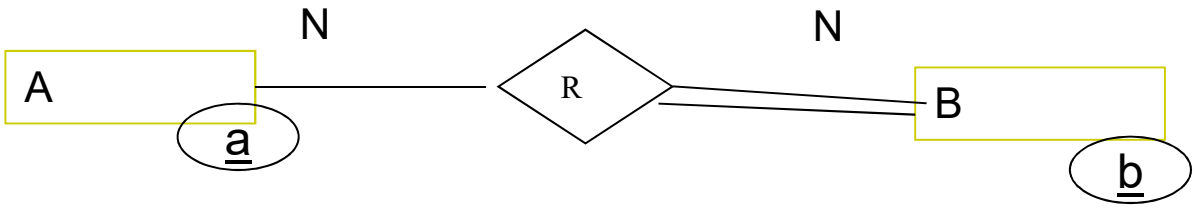


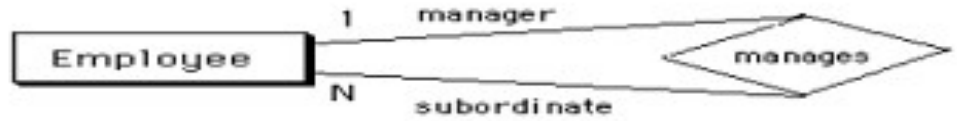
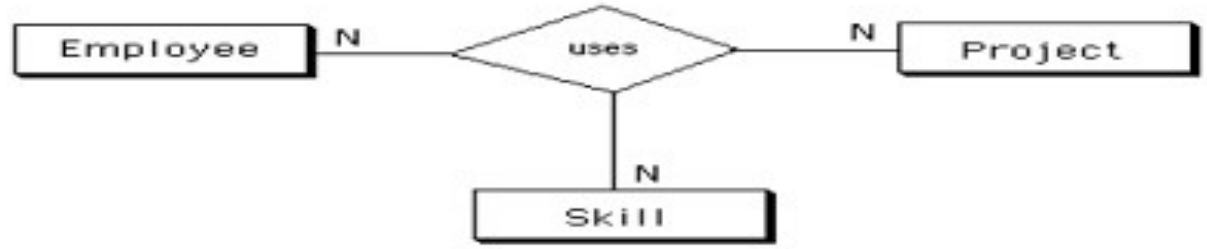


Var olma bağımlılığı *(existence dependency):*

- *B varlık kümesindeki her bir 'b' varlığı, A varlık kümesindeki bir 'a' varlığı ile mutlaka bir bağıntı kurması gerekiyorsa, «B varlık kümesi A varlık kümesine var olma bağımlıdır» denir. Burada A birincil varlık, B ikincil varlık olarak adlandırılır.*
- *(total / partial participation)*

ER:

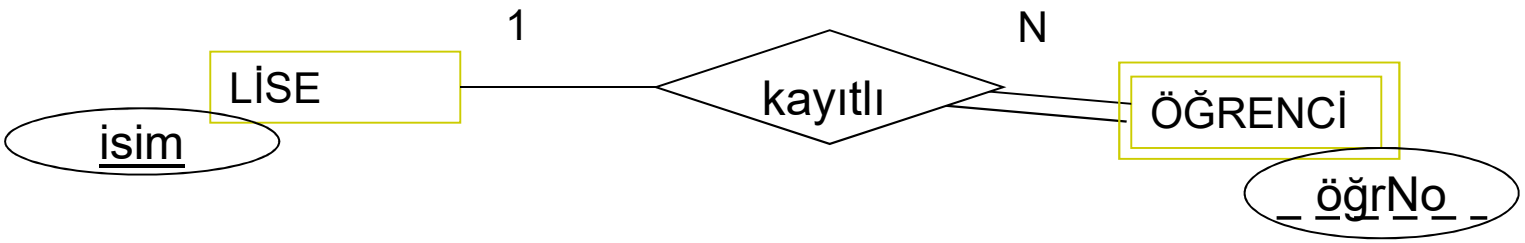


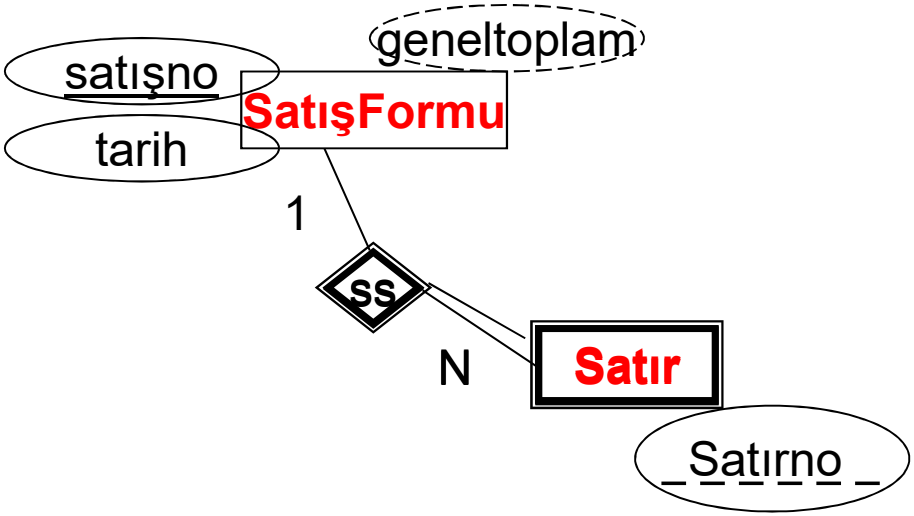
Concept	Representation & Example
Degree recursive binary	
binary	
ternary	
Connectivity one-to-one	
one-to-many	
many-to-many	
Existence optional	
mandatory	

Anahtar ve Güçlü/Zayif Varlık Kümeleri

- Varlık/Bağıntı kümesindeki varlık/bağıntıları birbirinden ayırd etmek için kullanılan nitelik veya nitelik grubuna **anahtar** denir.
  - **Super anahtar (*superkey*)**: Bütün anahtarlar super anahtardır.
  - **Aday anahtar (*candidate key*)** : **BA** Bir varlık/bağıntı **kümesinin 'K' super anahtarının hiç bir öz-alt kümesi super anahtar değilse**, 'K' o varlık/bağıntı kümesinin aday anahtarı olur. (diğer bir ifade ile **«aday anahtar»=«minimal super key»)**
- 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





Anahtar Belirleme

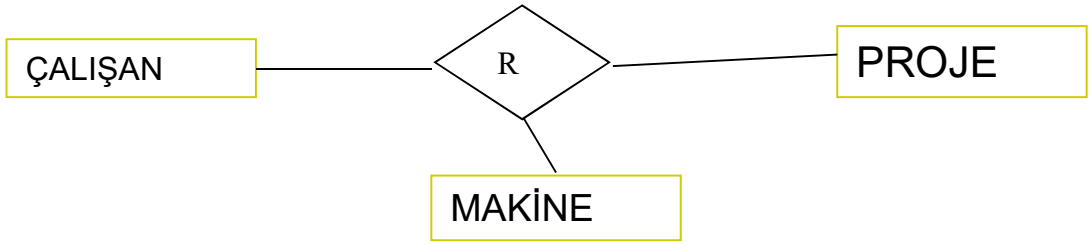
Bağıntılardaki anahtarlar:

- N-N bağıntı kümesinin anahtarı, bağıntı içerisindeki varlık kümelerinin anahtarlarının hepsidir.
- 1-N bağıntı kümesinin anahtarı N-tarafındaki bağıntının anahtarıdır.
- 1-1 bağıntı kümesinin anahtarı ise herhangi bir varlık kümesinin anahtarı olabilir.

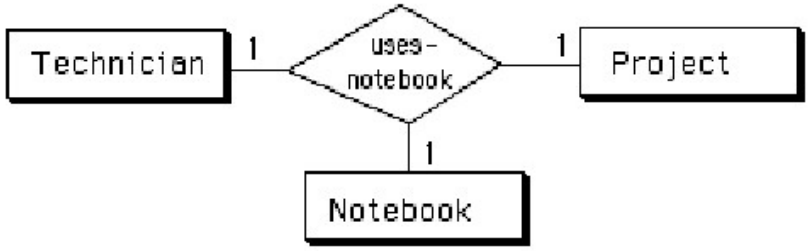
Zayıf varlık kümesinin anahtarı, var olma bağıntısı içerisindeki güçlü varlık küme(ler)sinin BA'ları + AN'dir.

- Buna göre; önceki örneklerdeki bağıntıların anahtarlarını belirleyiniz.

çok dereceli ER-bağıntı örnekleri



- Yukarıdaki 3'lü bağıntıda:
  - İki varlık arasındaki bağıntı, 3. varlıktan bağımsız bir şekilde ifade edilemiyor.



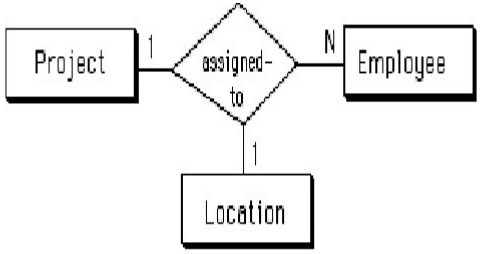
A technician uses exactly one notebook for each project. Each notebook belongs to one technician for each project. Note that a technician may still work on many projects and maintain different notebooks for different projects.

(a) one-to-one-to-one ternary relationship

Functional dependencies

emp-id, project-name -> notebook-no
emp-id, notebook-no -> project-name
project-name, notebook-no -> emp-id

çok dereceli ER-bağıntı örnekleri

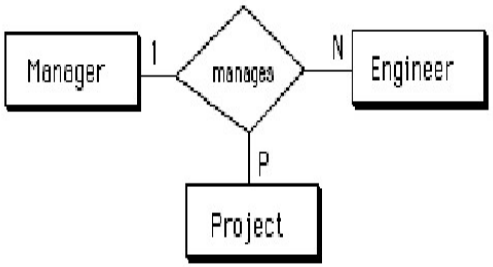


Each employee assigned to a project works at only one location for that project, but can be at different locations for different projects. At a particular location, an employee works on only one project. At a particular location, there can be many employees assigned to a given project.

Functional dependencies

emp-id, loc-name -> project-name
emp-id, project-name -> loc-name

(b) one-to-one-to-many ternary relationship

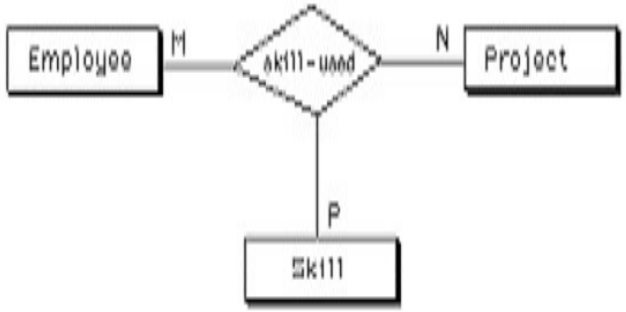


Each engineer working on a particular project has exactly one manager, but each manager of a project may manage many engineers, and each manager of an engineer may manage that engineer on many projects.

Functional dependency

project-name, emp-id -> mgr-id

(c) one-to-many-to-many ternary relationship



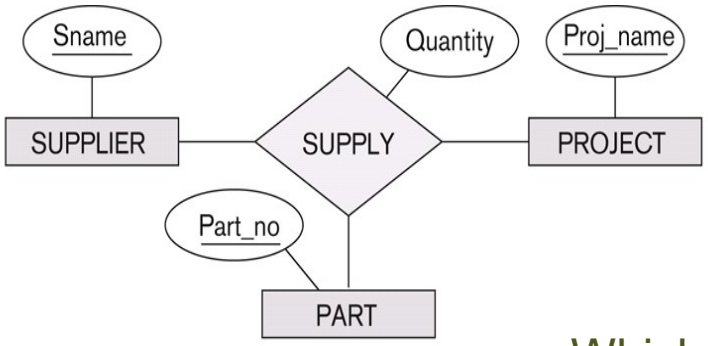
Employees can use many skills on any one of many projects, and each project has many employees with various skills.

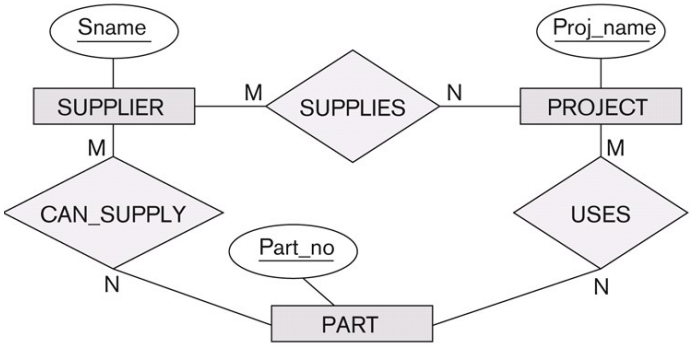
Functional dependencies

None

d) many-to-many-to-many ternary relationship

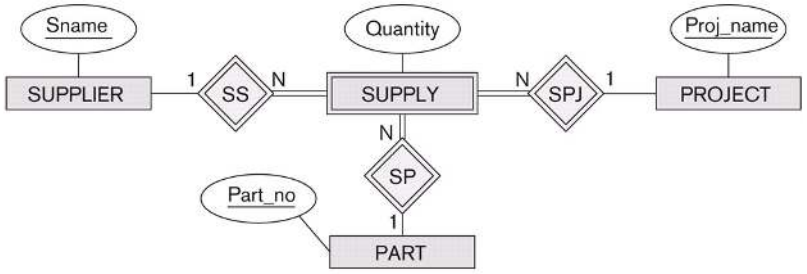
çok dereceli ER-bağıntı örnekleri



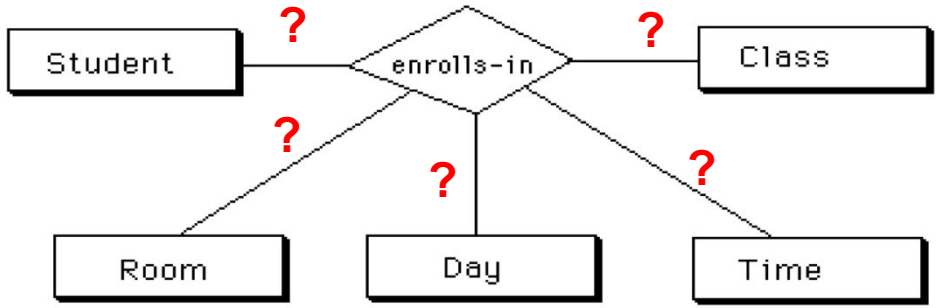


- Which one conveys more information?

  (s,j,p) ≠ (s,p) (j,p) (s,j)

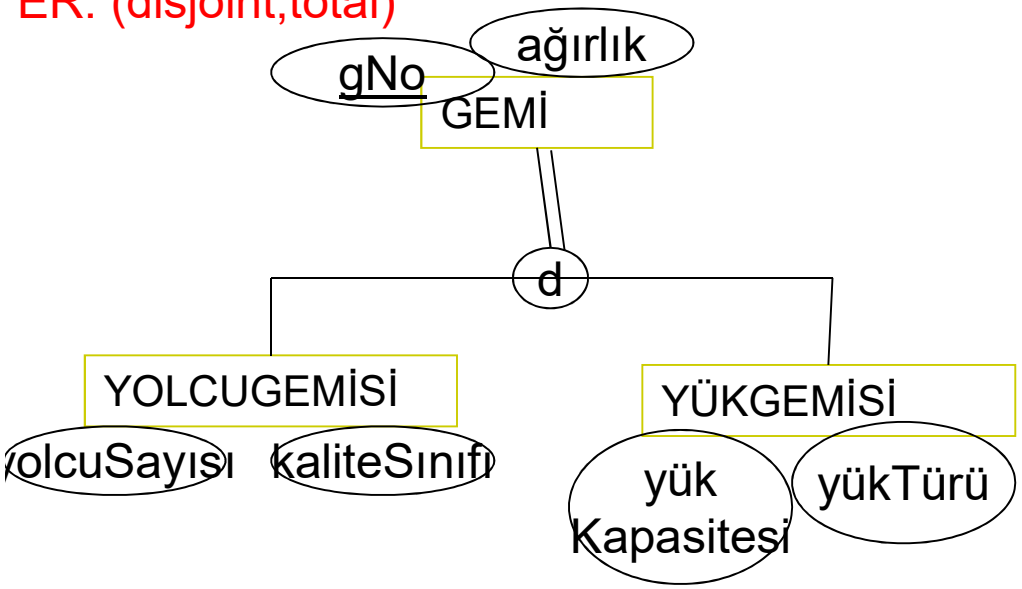


One more step further: If we do not want to use weak entity type, how to design?



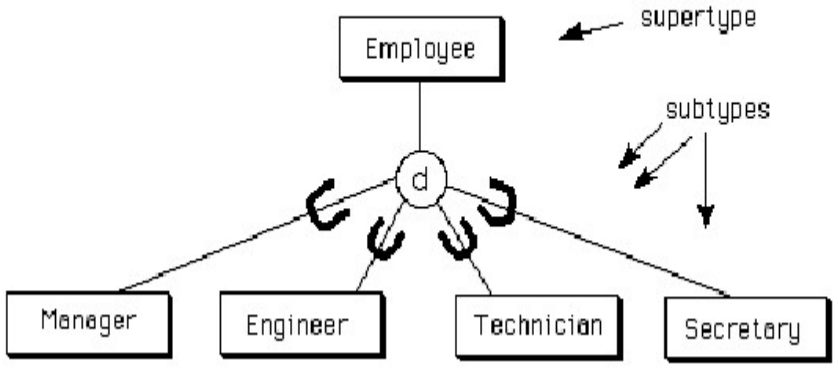
EER-Genelleme *(«IS A», «Ait olma» bağıntısı)*

ER: (disjoint,total)

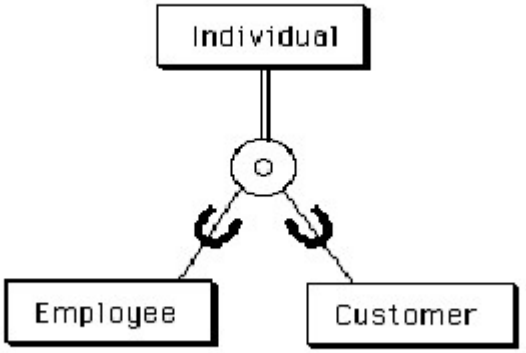


- Katılım kısıtlaması:
  - Zorunlu
  - Seçimli
- Ayrıklık kısıtlaması:
  - OR
  - AND
- 4 farklı kısıtlama tanımlanabilir, ihtiyaca göre kullanılabilir.
- **Supertype varlık nitelikleri/anahtarı, subtype varlıklarında geçerlidir.**

ER: (disjoint,partial)

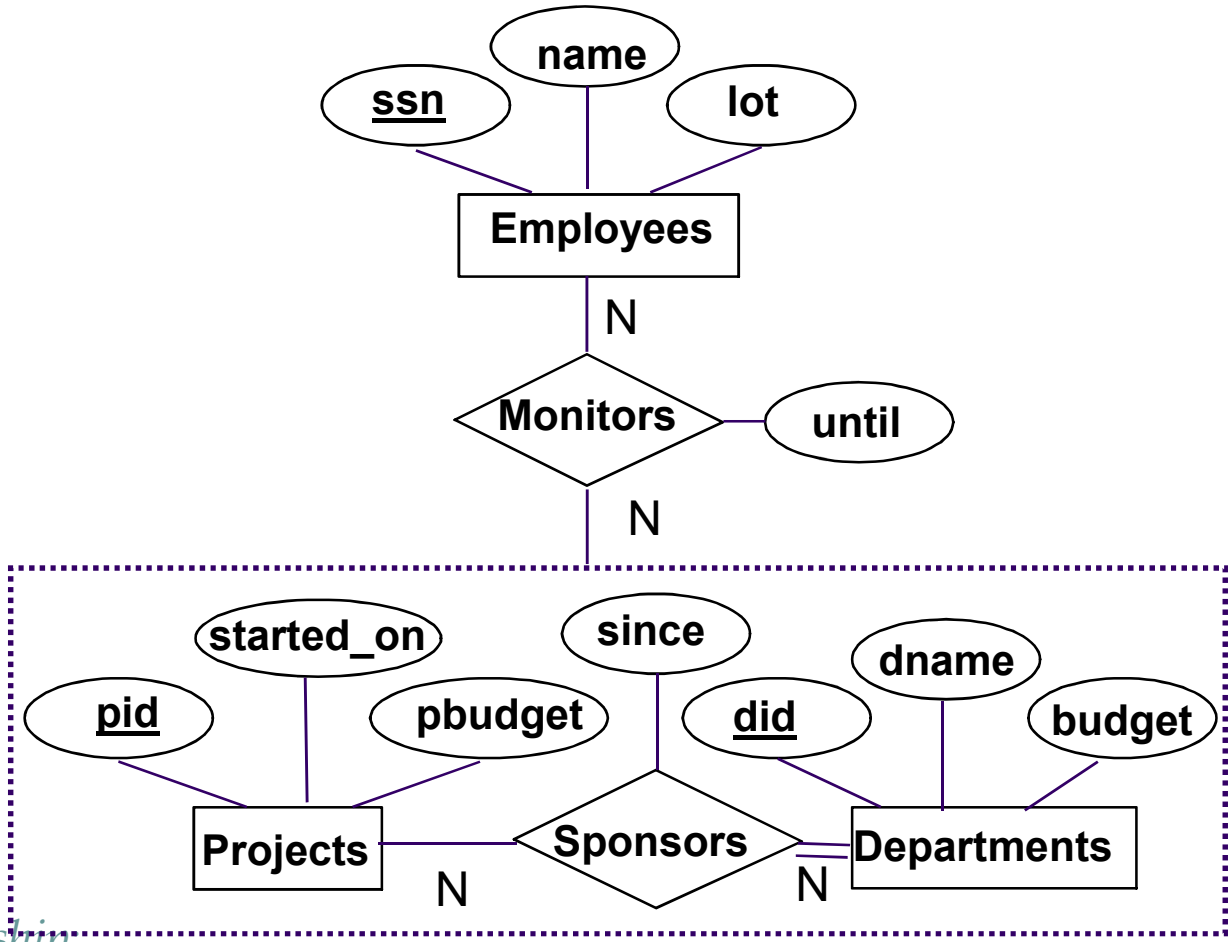


ER: (overlap,total)



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.



* *Aggregation vs. ternary relationship:*

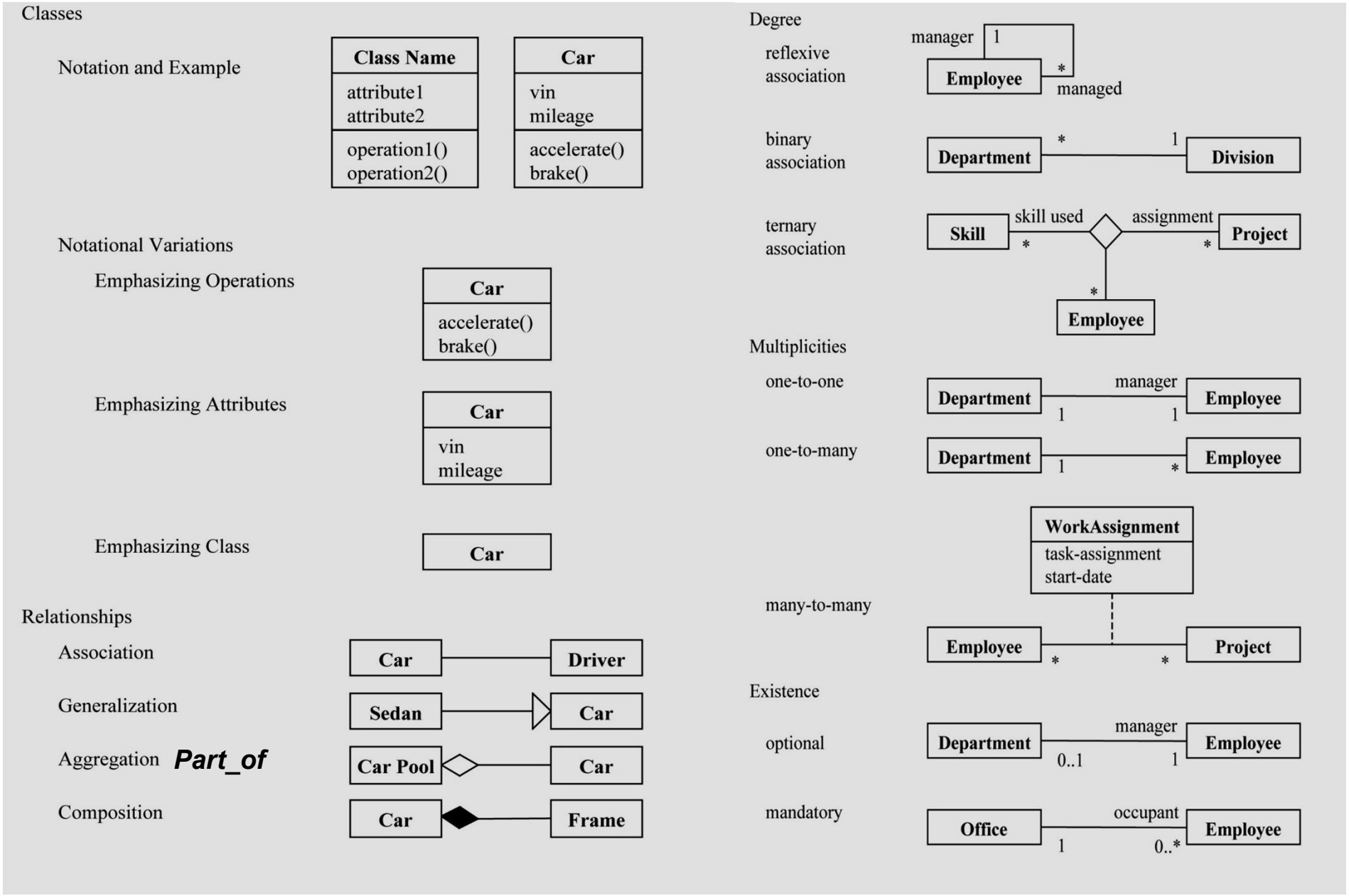
❖ Here, aggregation is best. Since «monitors» is a distinct relationship,with a descriptive attribute. Otherwise, if we had designed with ternary, «since» attribute will repeat for every «monitoring»

❖ Can we say that each sponsorship is monitored by at most one employee with ternary relationship / aggregation?

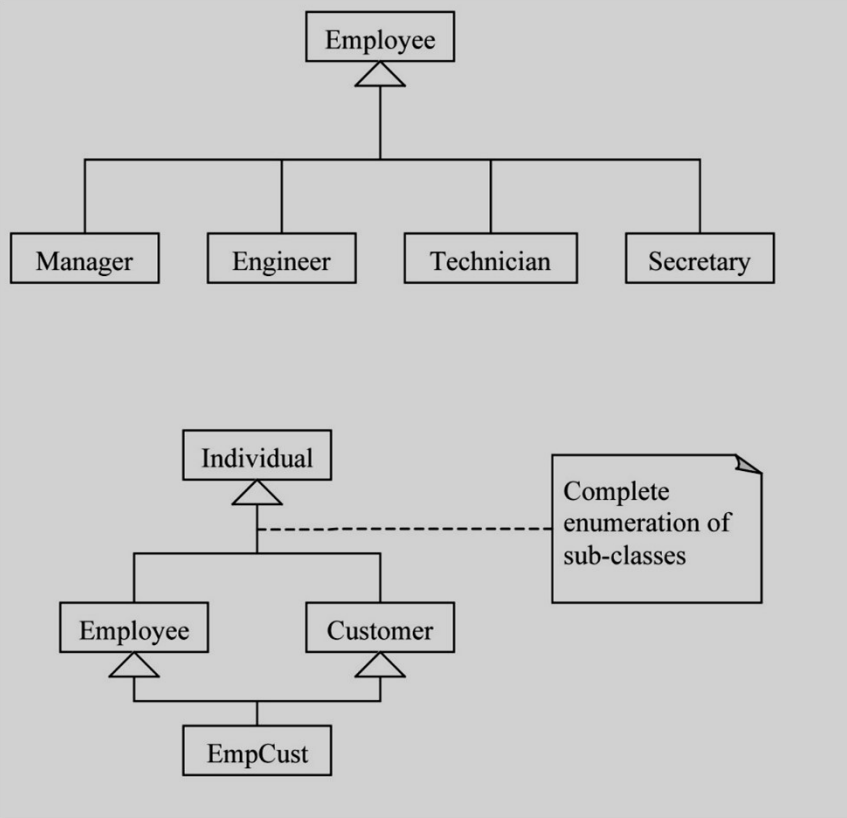
Always «Yes» with **aggregation**.

But **sometimes** with **ternary relationships**. Because, ER versions vary. In our ER version it is possible.

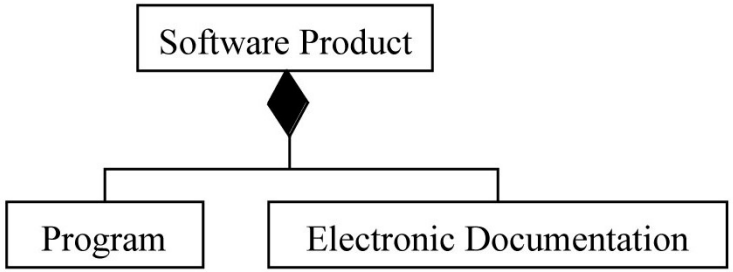
UML ile modelleme

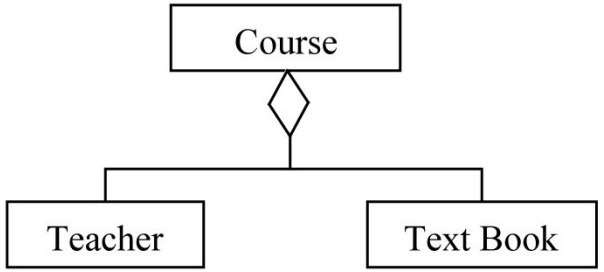


UML generalization constructs



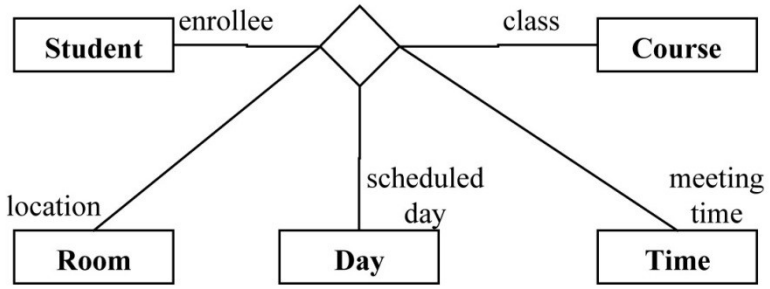
UML composition/aggregation constructs





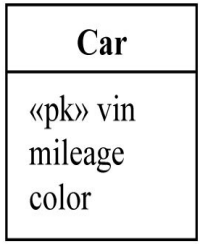
composition: parçalar tek başına bulunamaz, bir mana ifade etmiyorlar. Parça, kümeye varolma bağımlıdır.
aggregation: parçalar kendi başlarına bir mana ifade ediyorlar. Örneğin, Course silinse Teacher mevcut kalabilir.

UML *n*-ary relationship

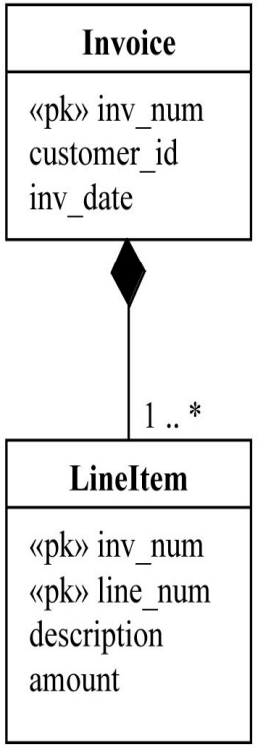


UML'de primary key

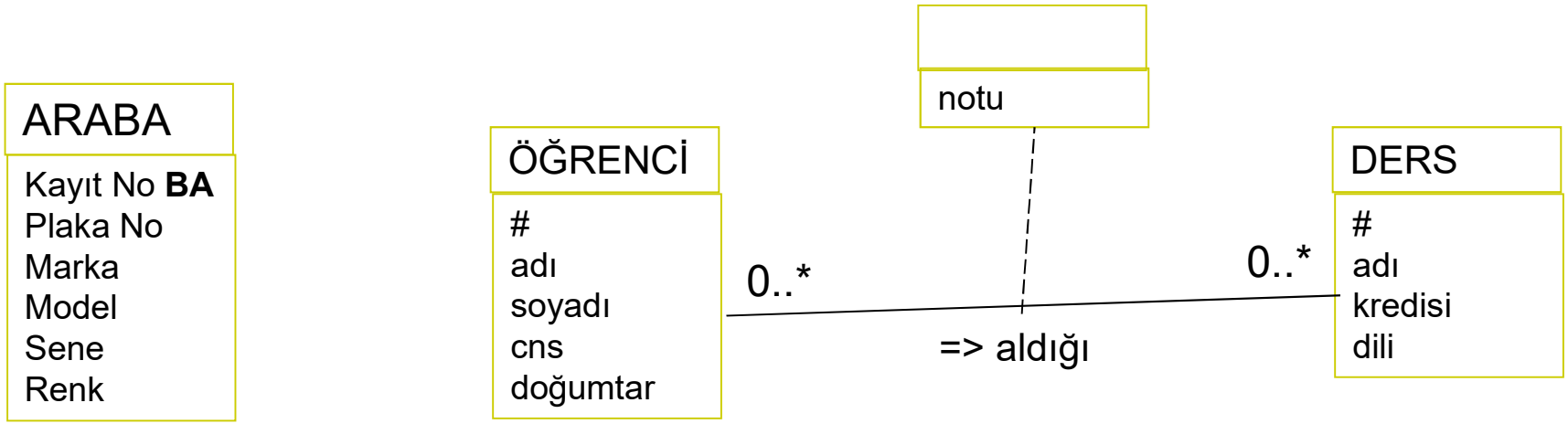
Primary key as a stereotype

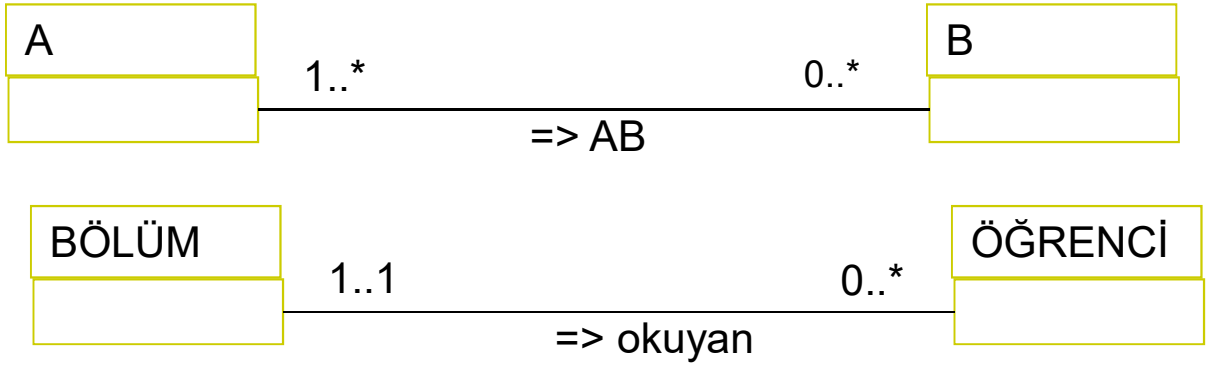


Composition example
with primary keys



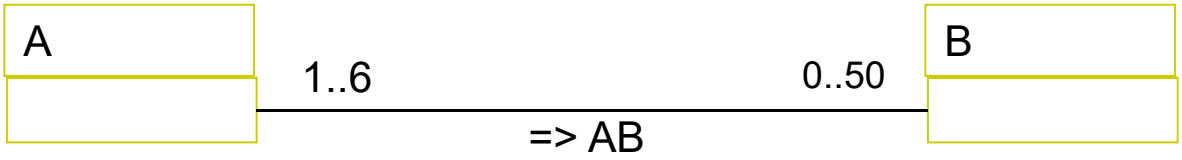
UML ile Varlık ve Bağıntı Örnekleri

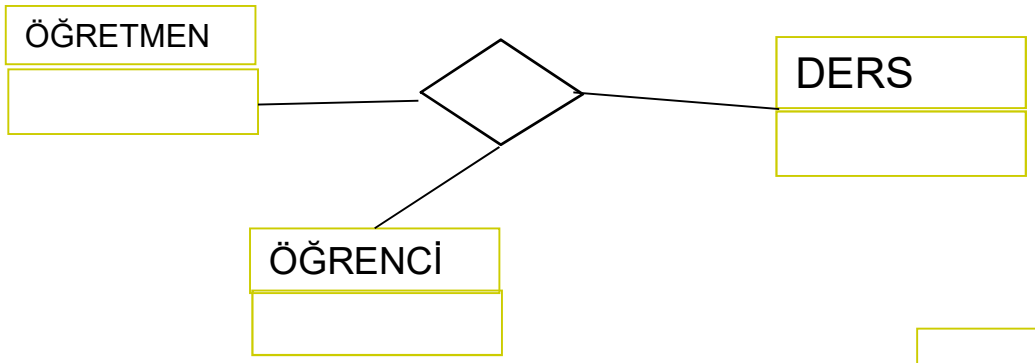


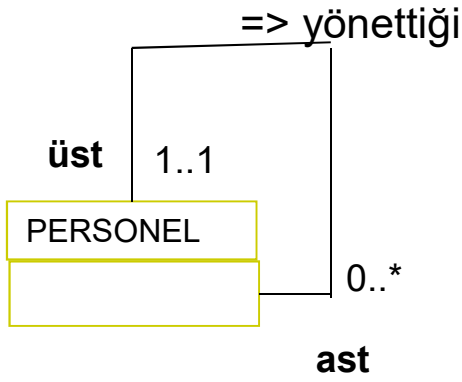


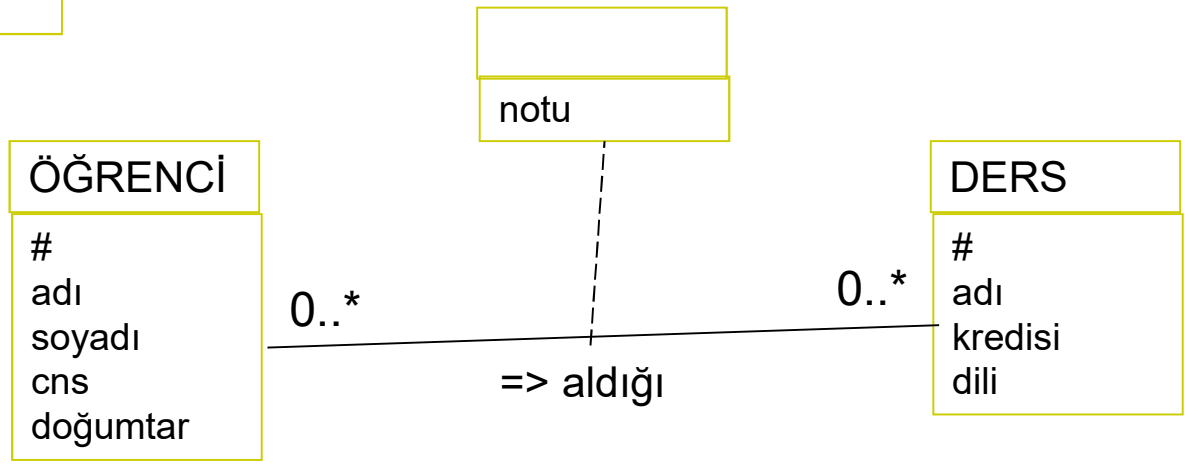
Örnek (UML)


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

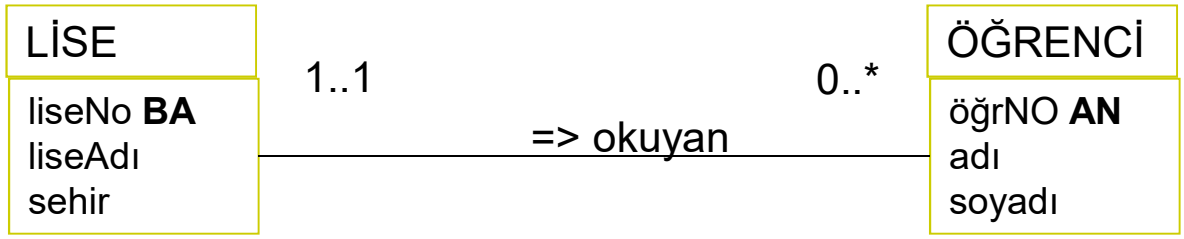

A
B
1..6
0..50
=> AB


ÖĞRETMEN
DERS
ÖĞRENCİ

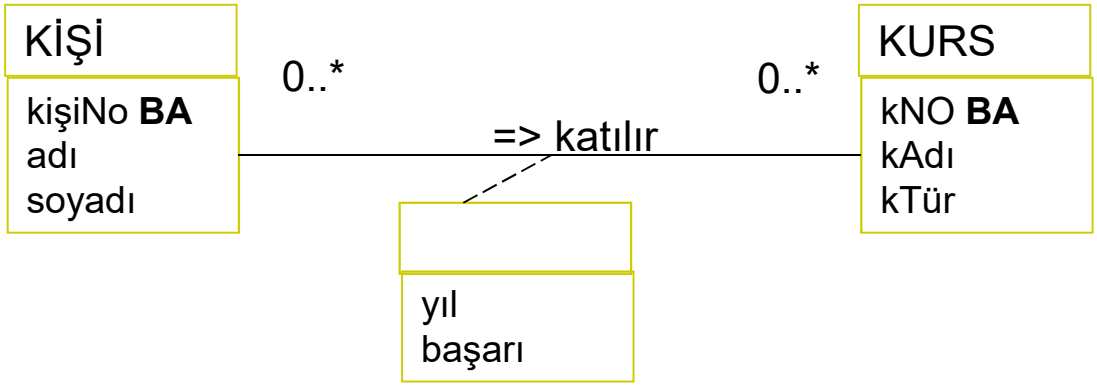

=> yönettiği
üst
1..1
PERSONEL
0..*
ast


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

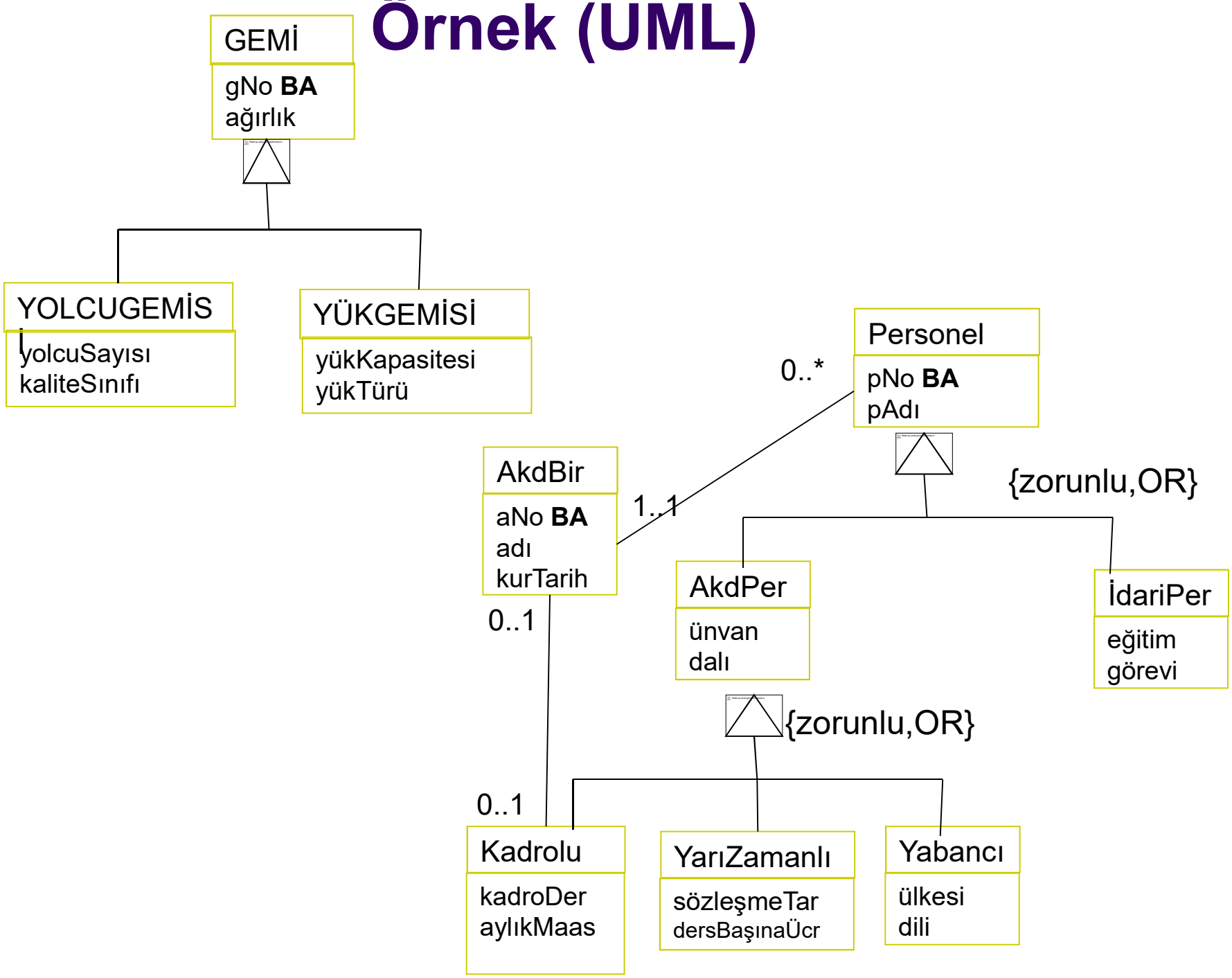
Örnek (UML)



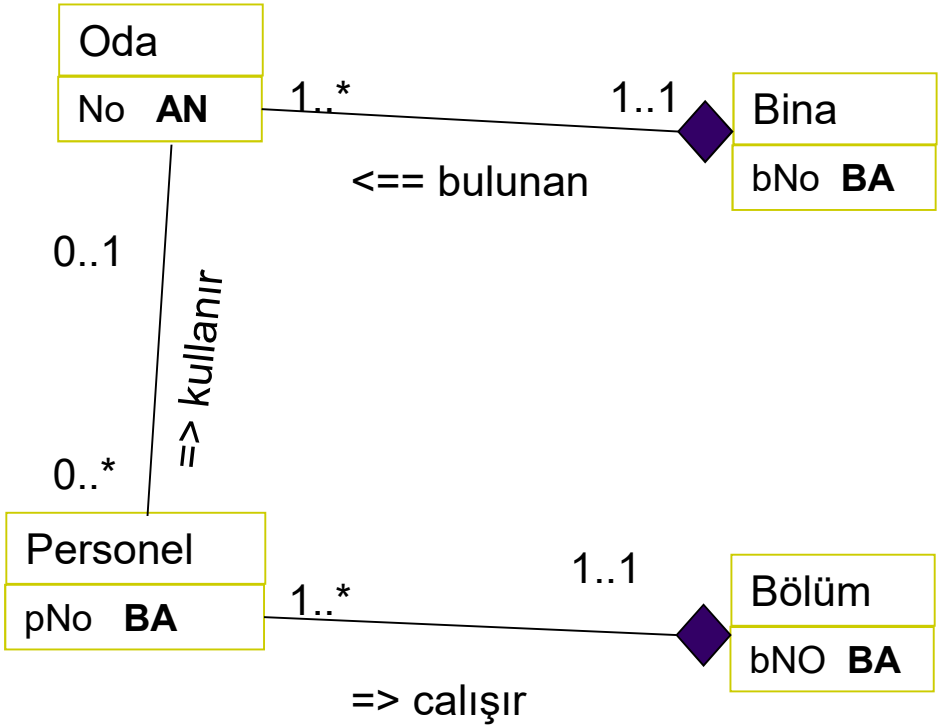
- Aşağıdaki örnekte bir kişi aynı kursu farklı yıllarda tekrar alabiliyorsa bağıntı kümesinin anahtarı ne olur?



Örnek (UML)


GEMİ
gNo **BA**
ağırlık
YOLCUGEMİS
yolcuSayısı
kaliteSınıfı
YÜKGEMİSİ
yükKapasitesi
yükTürü
Personel
pNo **BA**
pAdı
0..*
AkdBir
aNo **BA**
adı
kurTarih
1..1
{zorunlu,OR}
AkdPer
ünvan
dalı
İdariPer
eğitim
görevi
0..1
{zorunlu,OR}
0..1
Kadrolu
kadroDer
aylıkMaas
YarıZamanlı
sözleşmeTar
dersBaşınaÜcr
Yabancı
ülkesi
dili

UML, composition



Örnek 1: COMPANY Database

- We need to define a database schema design 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

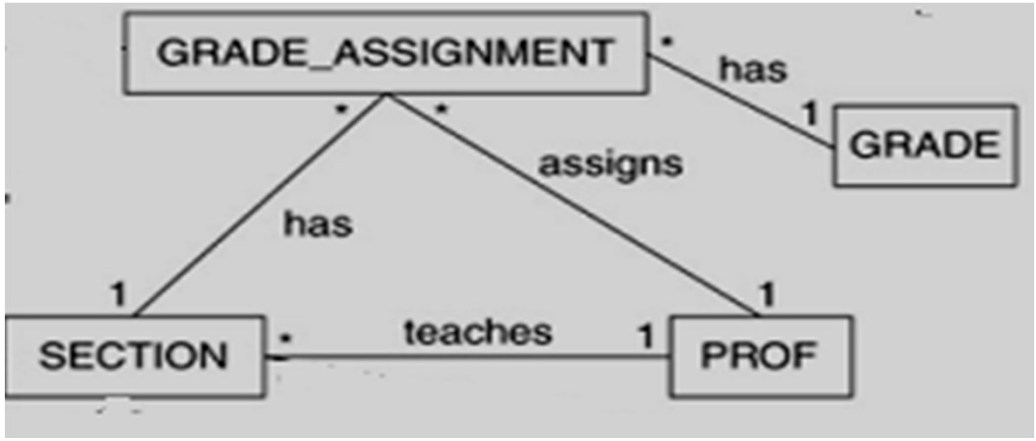
İlk tasarım sonrası iyileştirmeler-1

- *Eğer mevcut nitelik başka bir varlığa işaret ediyorsa:*
  - Nitelik➔ Bağıntı haline dönüsmeli.
- *Eğer mevcut nitelik cok 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. Yukarda-aşağıya (top-down), aşağıdan-yukarıya (bottom-up) yaklaşımlar kullanılmalı.
- Bağıntıların dereceleri tekrar degerlendirilmeli, gerekirse "kümeleme" ile daha acık yapılanmalar kullanılmalı.

İlk tasarım sonrası iyileştirmeler-2:

fazlalıkların çıkarılması

- *Eğer bir bağıntı diagramdan cıkarıldığı zaman bilgi kaybı olmuyorsa, bu bağıntının fazladan/gereksiz (redundant) olduğuna karar verilmiş olur. Genelde, diagramdaki cevrimler kontrol edilir. Cevrim icindeki bir bağıntının fazladan olma ihtimali vardır.*
  - Aşağıdaki örnekte, bir SECTION'a ait not girişi takip ediliyor. Eğer girilen bir not (GRADE), bir SECTION için ise ve o SECTION bir PROF tarafından veriliyosa; "assigns" bağıntısına ihtiyaç olacak mıdır? Elbette "Olmayacaktır".

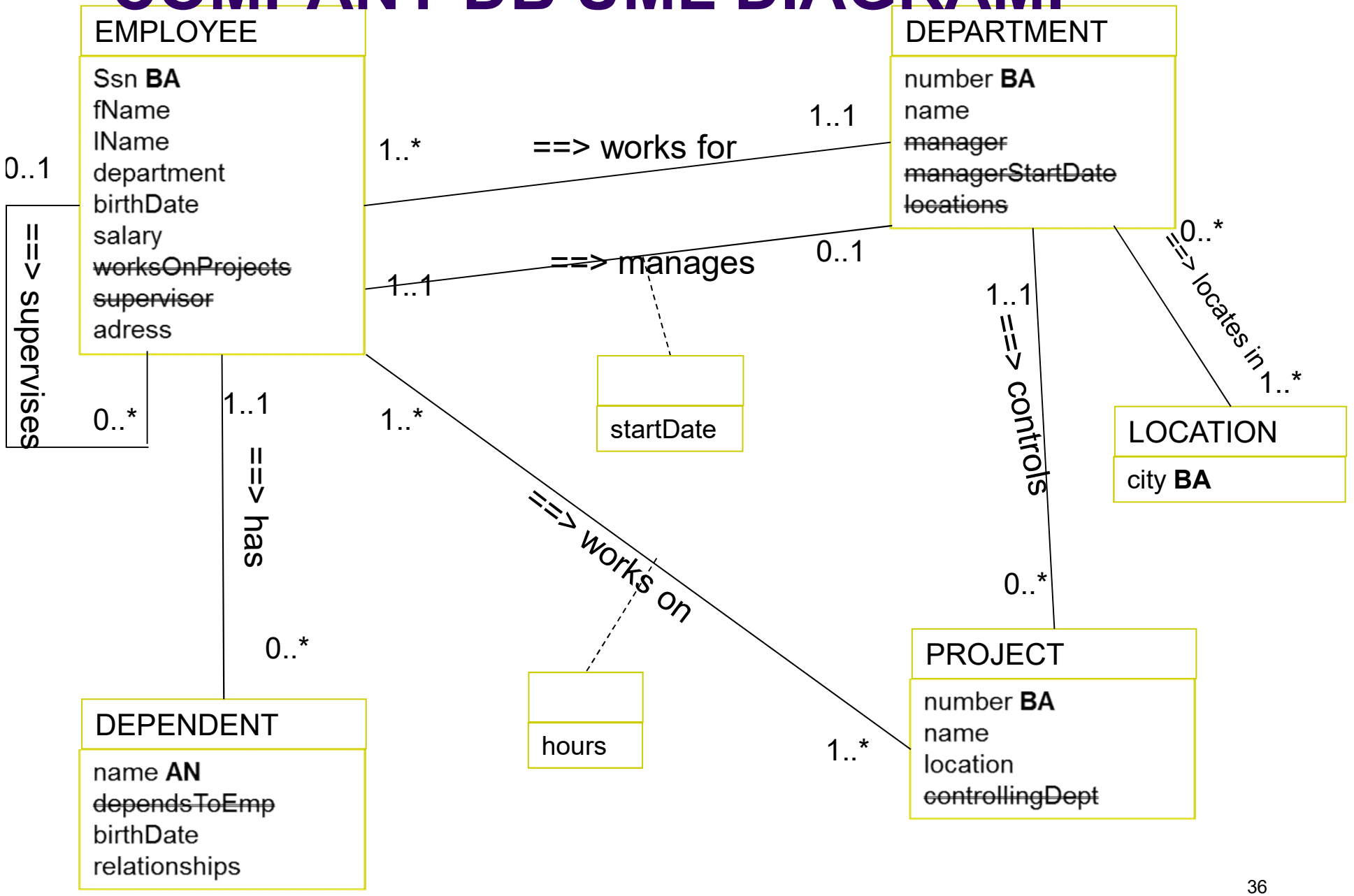


COMPANY DB (devam)

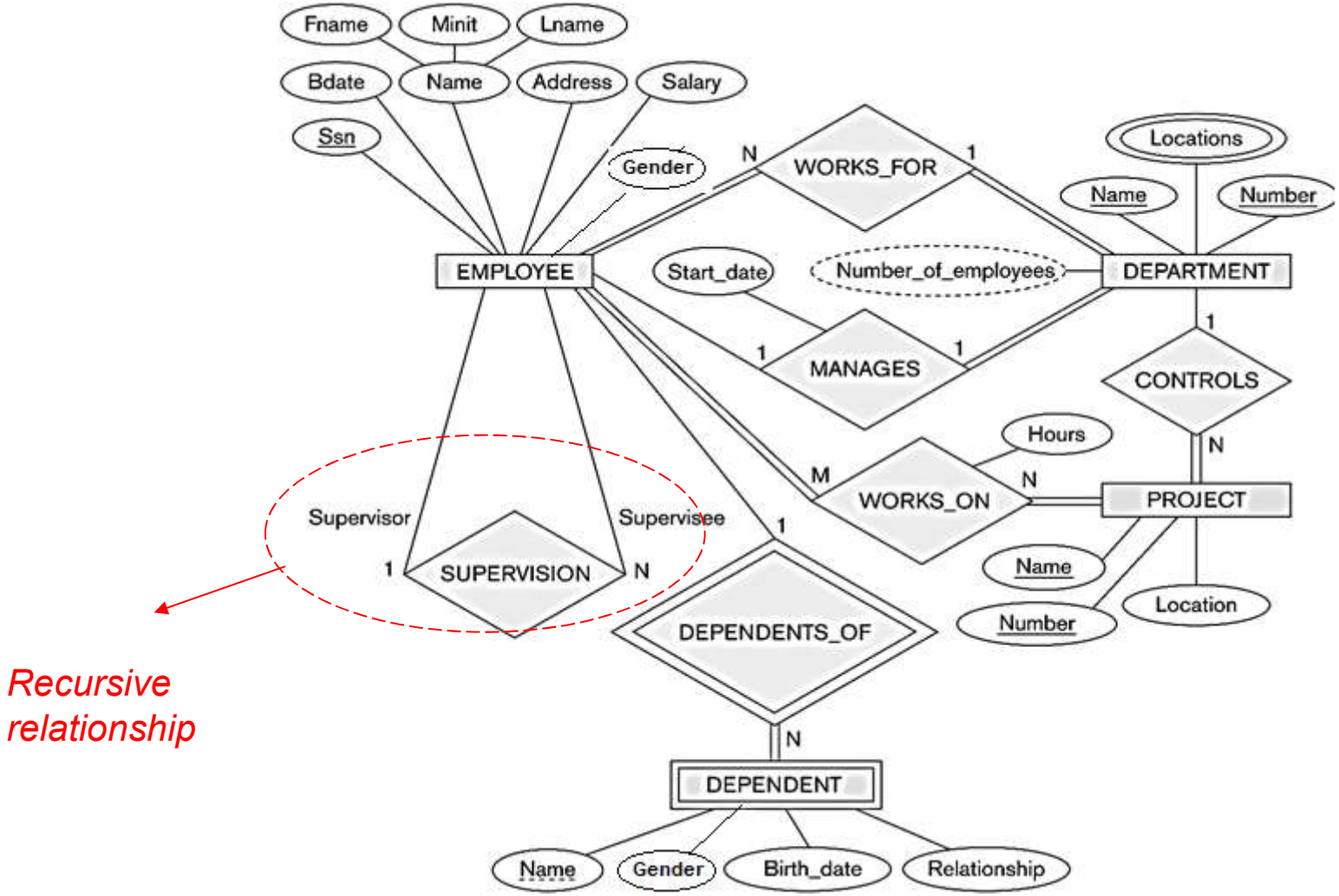
- In the refined design, some attributes from the initial entity types are refined into relationships:
  - Manager of DEPARTMENT -> MANAGES
  - Works_on of EMPLOYEE -> WORKS_ON
  - Department of EMPLOYEE -> WORKS_FOR
  - Controlling_Department of PROJECT → CONTROLS
  - Supervisor of EMPLOYEE → SUPERVISION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

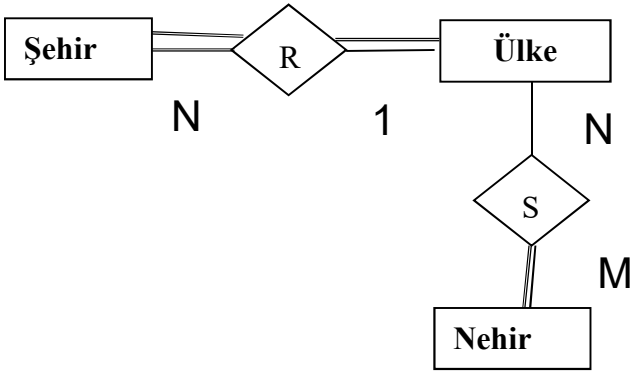


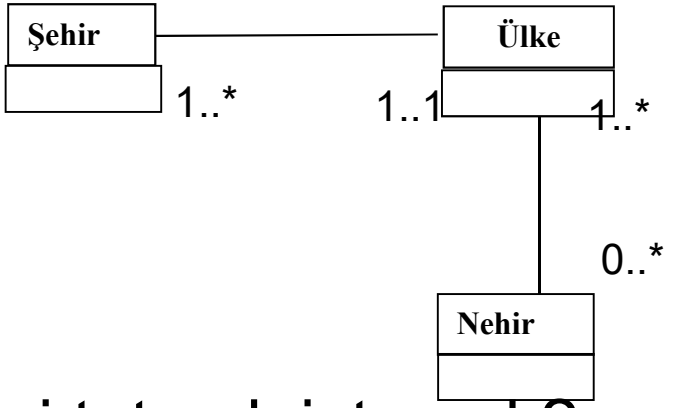
COMPANY DB ER DIAGRAM



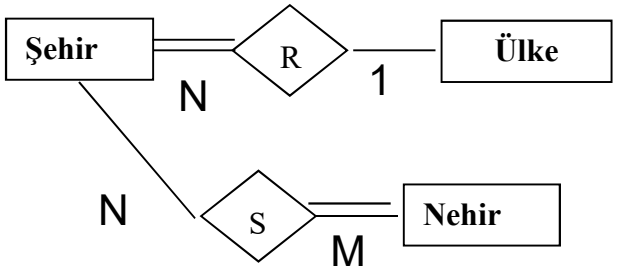
Örnek-2 (ER,UML)

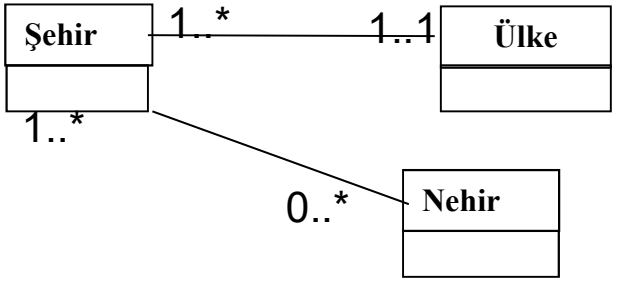
- Şehirler, ülkeler ve nehirleri gösteren bir veritabanı tasarımı.





- Nehirlerin hangi şehirlerden geçtiğini tutmak istersek?





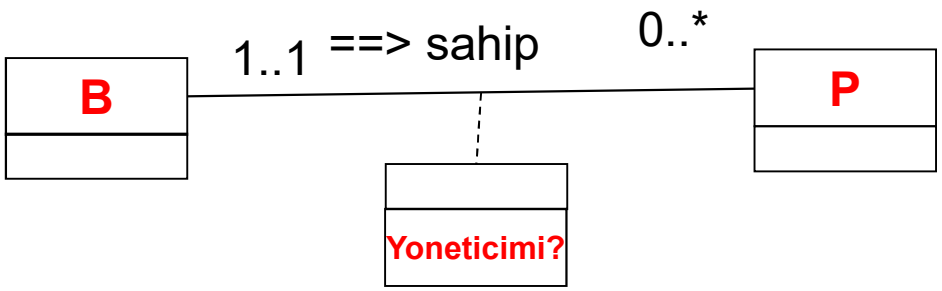
Örnek 3

Bir şirketteki bölümler ve personel bilgilerini tutan Bölüm-Personel veri tabanını düşünelim.

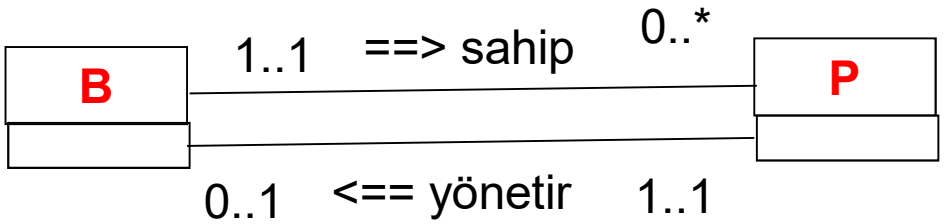
- Her **bölüm** çok sayıda **personele** sahiptir ve her personel en fazla bir bölüme bağlıdır.
- Bir bölümün sadece bir yöneticisi (*mutlaka*) vardır.

Buna göre aşağıdaki durumları *birbirinden bağımsız olarak* cevaplayınız…

- Bir bölümün yöneticisi ancak o bölüme bağlı bir personel olabiliyor ise, Nasıl bir ER modeli çizilebilir?



- Bir bölümün yöneticisi herhangi bir personel olabiliyor ise, Nasıl bir ER modeli çizilebilir?

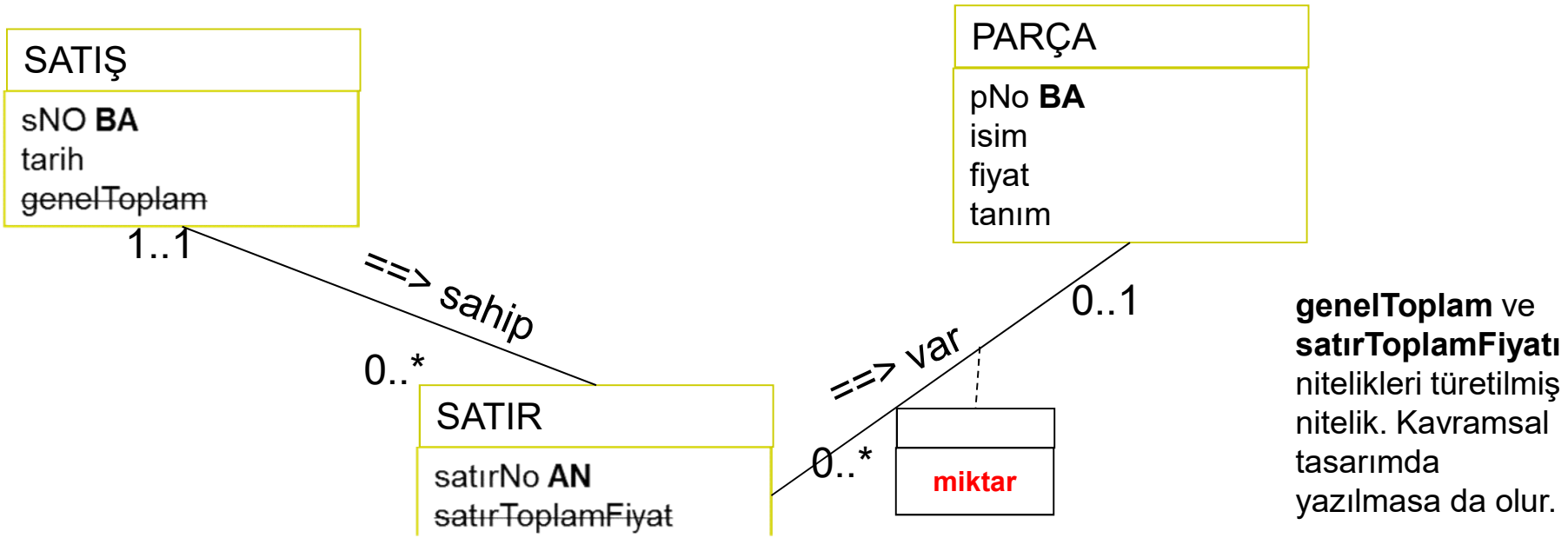


Örnek 4

		SATIŞ FORMU				
SatışNO: 12						**Tarih:** 11.06.2009
SatırNo	**ParÇaNo**	**ParÇaİsim**	**Tanım**	**Miktar**	**Birim Fiyat**	**Toplam**
1	23645	ayakkabı	Spor,basketbol	1	110	110
2	12674	gomlek	klasik tarz	2	80	160
3	...		...			
					GenelToplam:	**880**

Yukarıda satış bilgileri tutan örnek bir satış formu içeriği görülmektedir. Buna karşılık gelen veri tabanı, **SATIŞ, PARÇA ve SATIR** olmak üzere 3 varlık setinden oluşmaktadır. (Aynı ParçaNolu parçalar farklı satırlarda tekrar edebilir.)

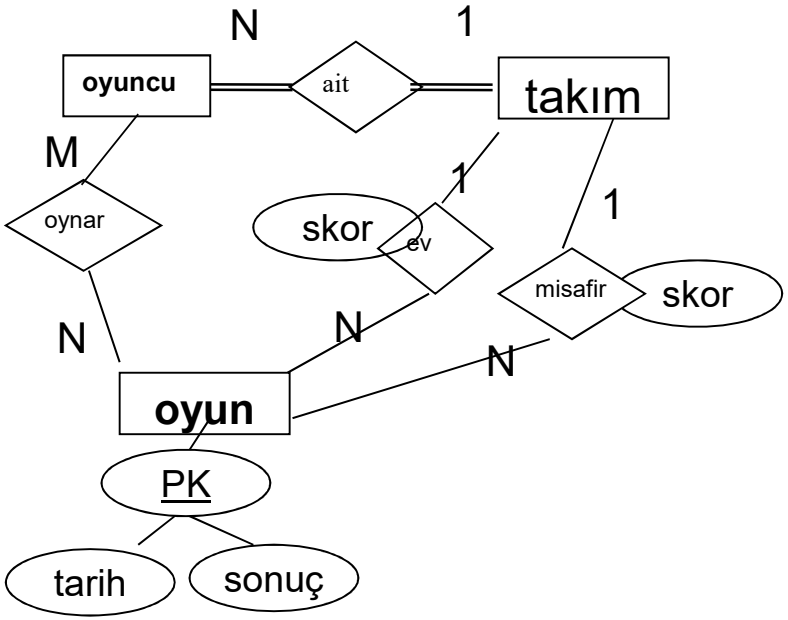
Bu varlık setlerini, özeliklerini ve aralarındaki ilişkileri gösteren ER diyagramını çiziniz.



genelToplam ve **satırToplamFiyatı** nitelikleri türetilmiş nitelik. Kavramsal tasarımda yazılmasa da olur.

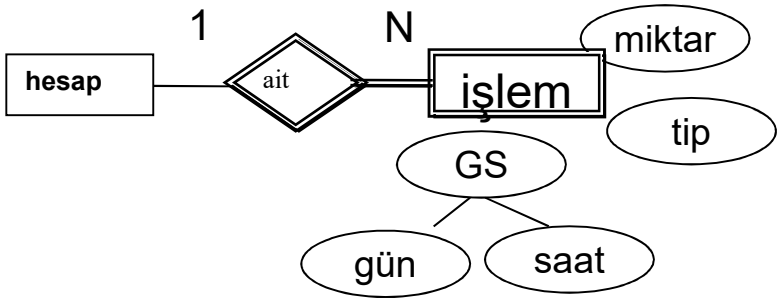
Örnek 5

- Birden çok oyuncu içeren takımlar arasında, iki takım içeren oyunlar ile ilgili bilgiler tutulmak isteniyor. *(Oyuncu en fazla bir takımın elemanı olabilir.)*
- her oyunda hangi takımların yer aldığı (hangi takım ev sahibi hangisi misafir olduğu) ve oyunun tarihi ve sonucu gibi bilgiler tutuluyor.
- her oyunda takımın hangi oyuncularının yer aldığı bilgisi takip ediliyor.



Örnek 6

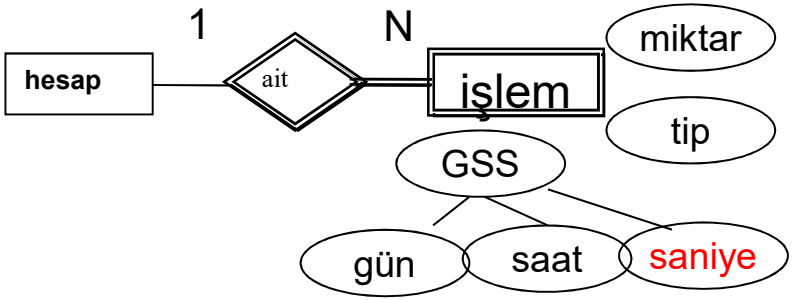
- Bir banka hesabına ait işlemler hakkında işlem zamanı(*gün,saat*), işlem tipi ve miktar bilgileri tutuluyor.bu olayı takip eden ER diyagramı:



Not: gün,saat partial key oluyor.

Farklı hesaplarda aynı saat içinde aynı miktar ve tip işlem olabilir!!!

- Yukarıdaki çözüm, aynı hesap üzerinde aynı saat içinde işlem yapılmaması durumunda geçerlidir. Bu koşul kalktığı zaman, tasarım işlem zamanı olarak sistem zamanını saklamalıdır;



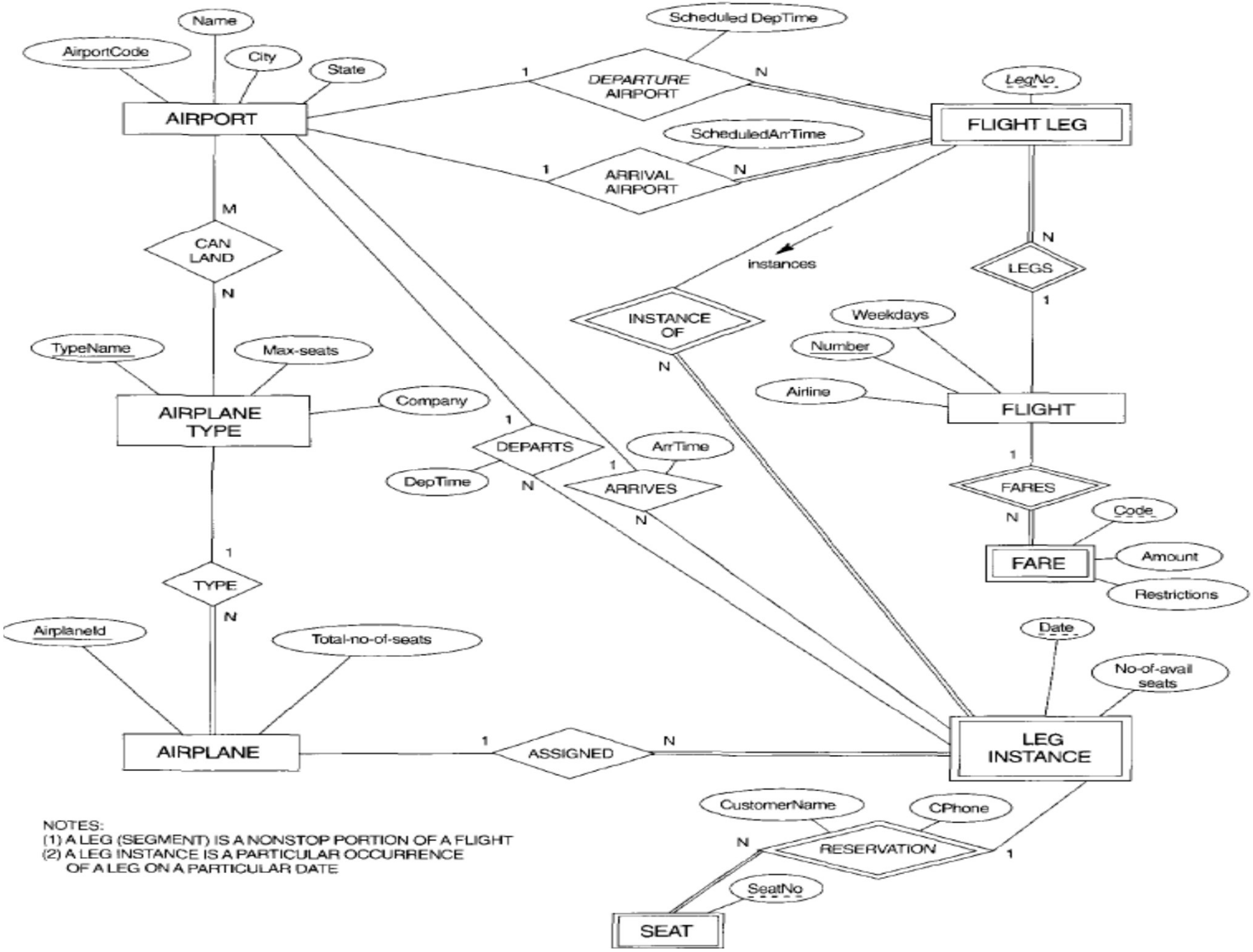
Not: gün,saat,saniye partial key oluyor.

Farklı hesaplarda aynı saniye içinde aynı miktar ve tip işlem olabilir!!!

UÇAK REZERVASYON SİSTEMİ

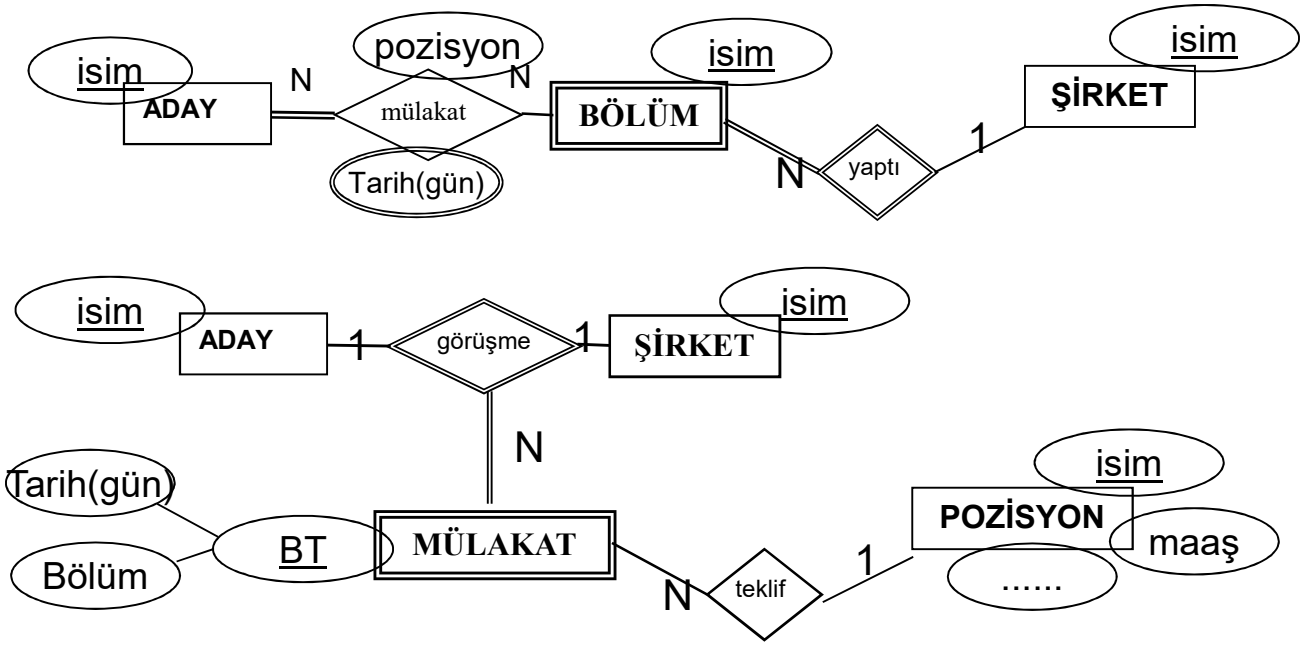
- Each FLIGHT is identified by a flight NUMBER, and consists of one or more FLIGHT_LEGs with LEG_NUMBERs 1, 2, 3, etc.
- Each leg has scheduled arrival and departure times and airports, and has many LEG_INSTANCEs--one for each DATE on which the flight travels. FARES are kept for each flight.
- For each leg instance, SEAT_RESERVATIONs are kept, as is the AIRPLANE used in the leg, and the actual arrival and departure times and airports.
- An AIRPLANE is identified by an AIRPLANE_ID, and is of a particular AIRPLANE_TYPE.
- CAN_LAND relates AIRPLANE_TYPEs to the AIRPORTs in which they can land.
- An AIRPORT is identified by an AIRPORT_CODE.

Örnek 7: Figure 3.17, Elmasri/Navathe book



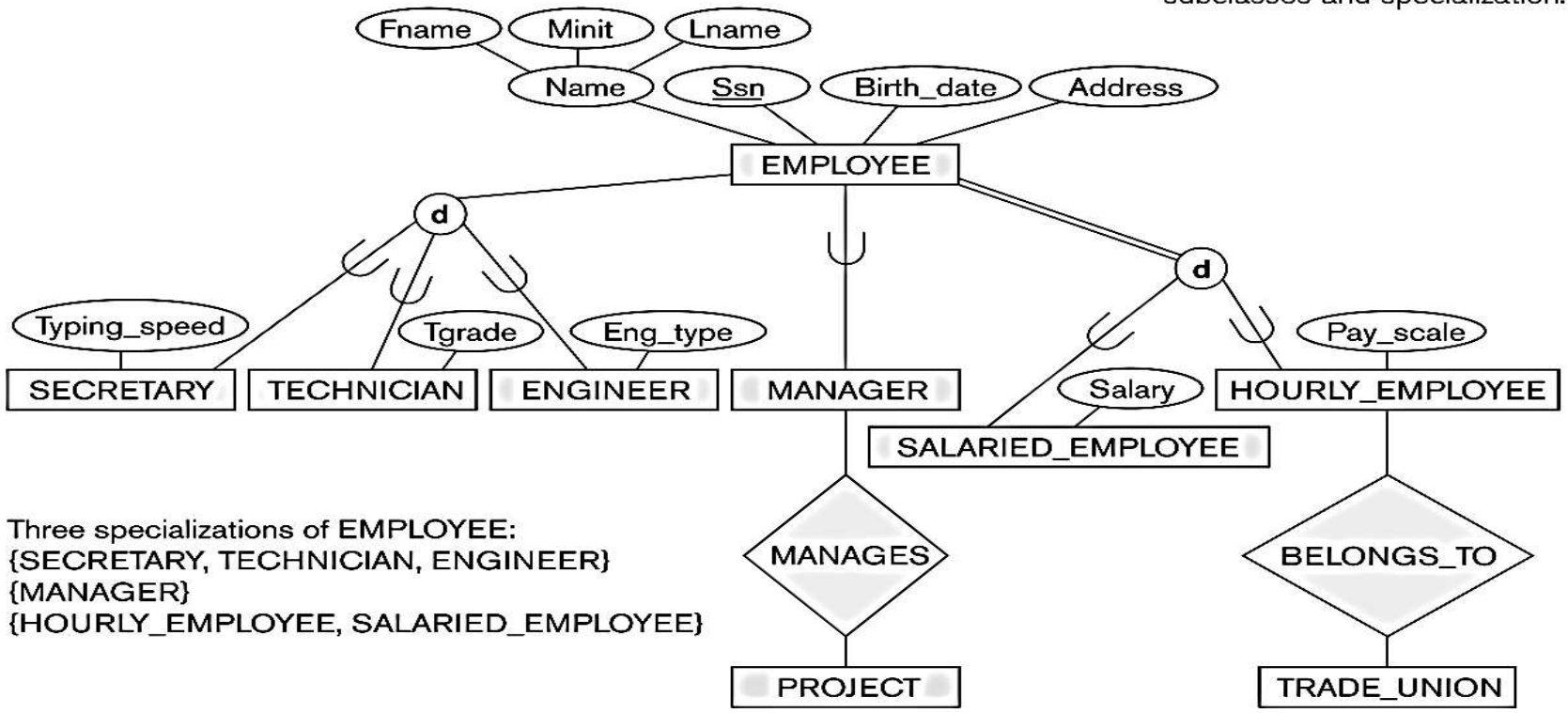
Örnek 8

- Adayların şirketler ile yaptıkları görüşme ve teklif edilen iş pozisyonunu takip etmek istiyoruz.
  - Aday bir şirketin birden çok bölümü ile görüşme yapabilir ve her bölümden farklı pozisyon teklifler alabilir; bir bölümden ancak 1 pozisyon teklifi alabilir.
  - Teklif edilen pozisyonun maasi, özellikleri gibi bir takım özellikler daha tutulmak isteniyor..
  - Görüşmenin yapıldığı gün (yıl/ay/gün) bilgisi tutluyor. Aynı bölüm ile görüşme ancak farklı bir günde olabiliyor.
  - Gerek aday isimleri gerek şirket isimleri sistemde biricik olduğunu varsayabiliriz..
  - Bölüm hakkında bir bilgi tutulmuyor.



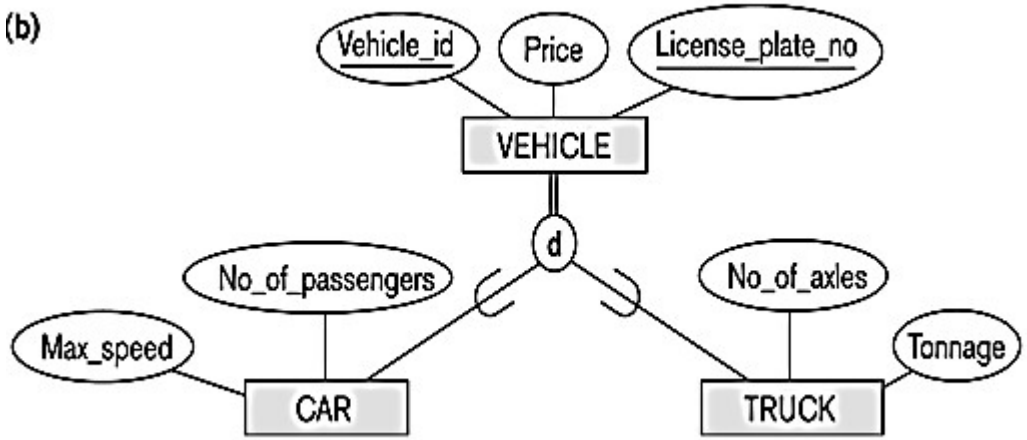
Örnek 9 (genelleme)

Figure 4.1
EER diagram notation to represent subclasses and specialization.



- ER diyagramında IS A (AİT OLMA) bağıntısı
  - Katılım, zorunlu ise çift çizgi ile, zorunlu değilse tekçizgi ile
  - Ayrıklık, AND(*disjoint*) ise yuvarlak içinde **d** ile, OR (*notdisjoint*) ise yuvarlak içinde **o** ile ifade eidliyor.

Örnek 10 (genelleme)



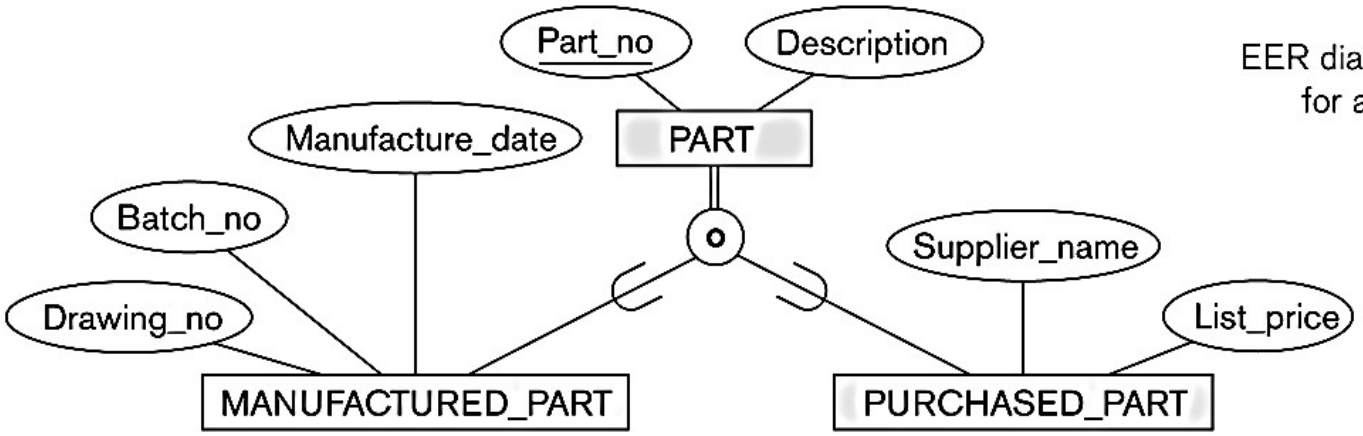


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Örnek 11 (genelleme)

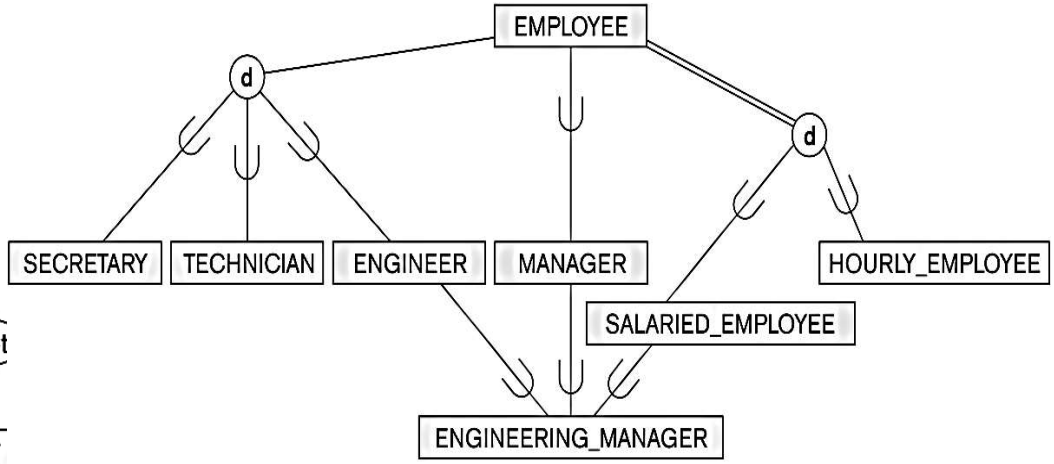


Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.

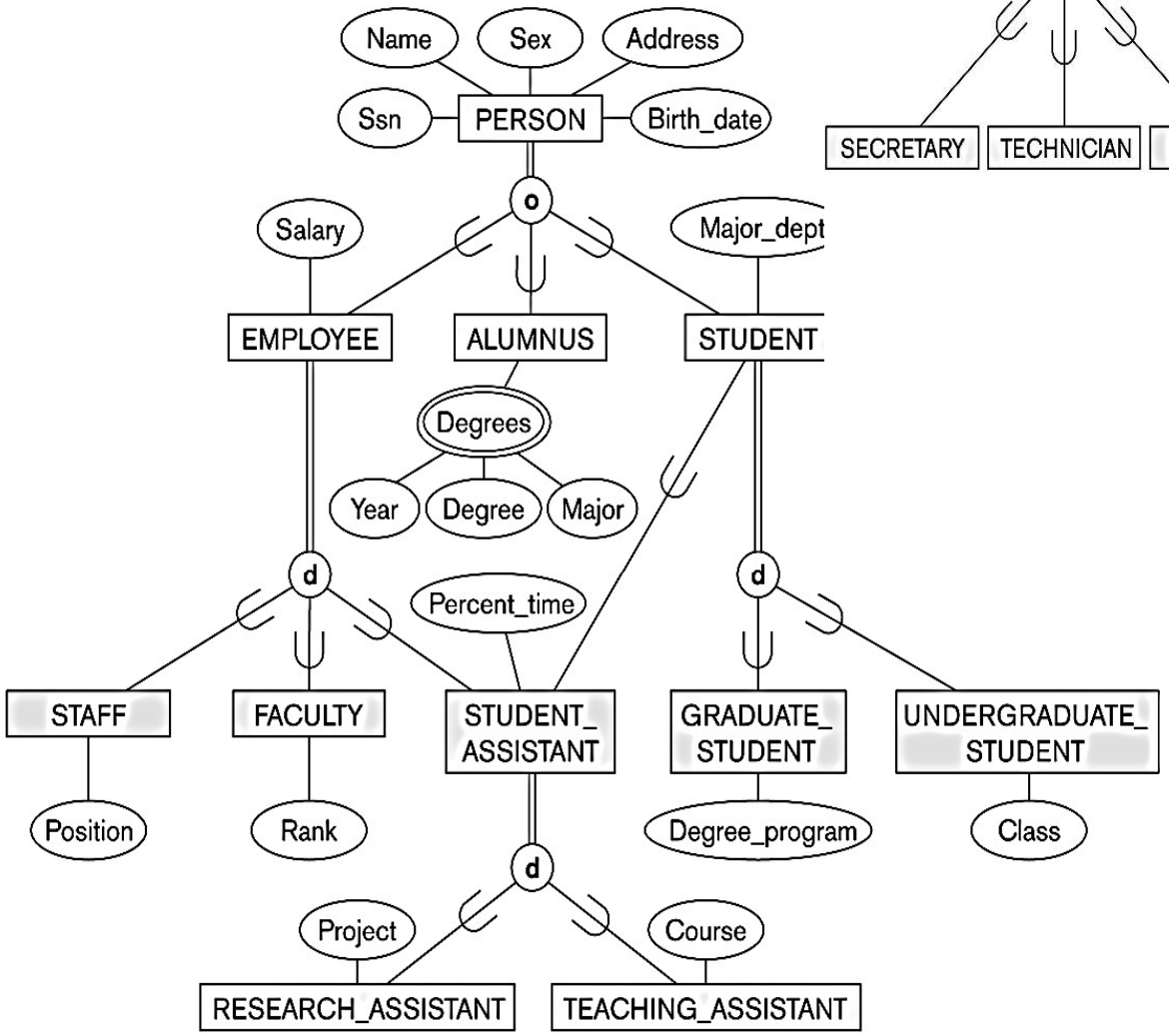
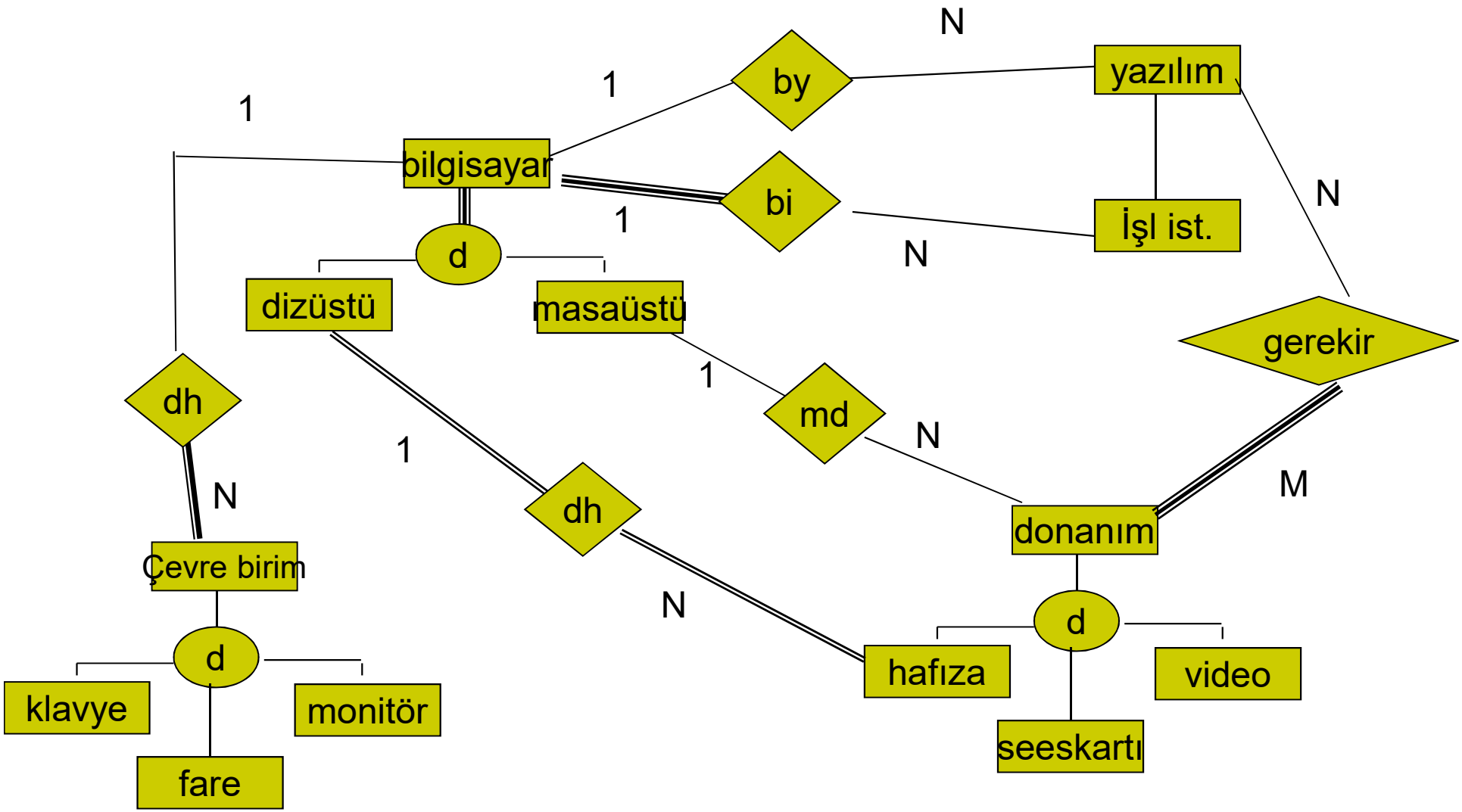


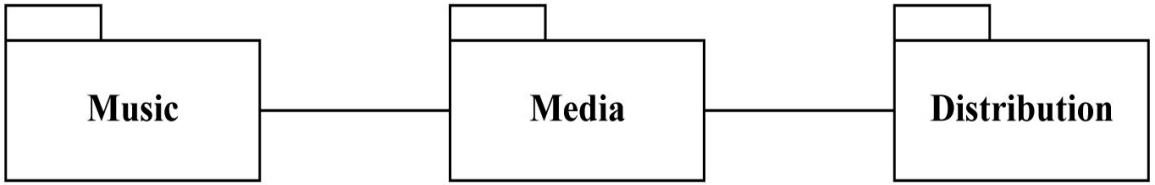
Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

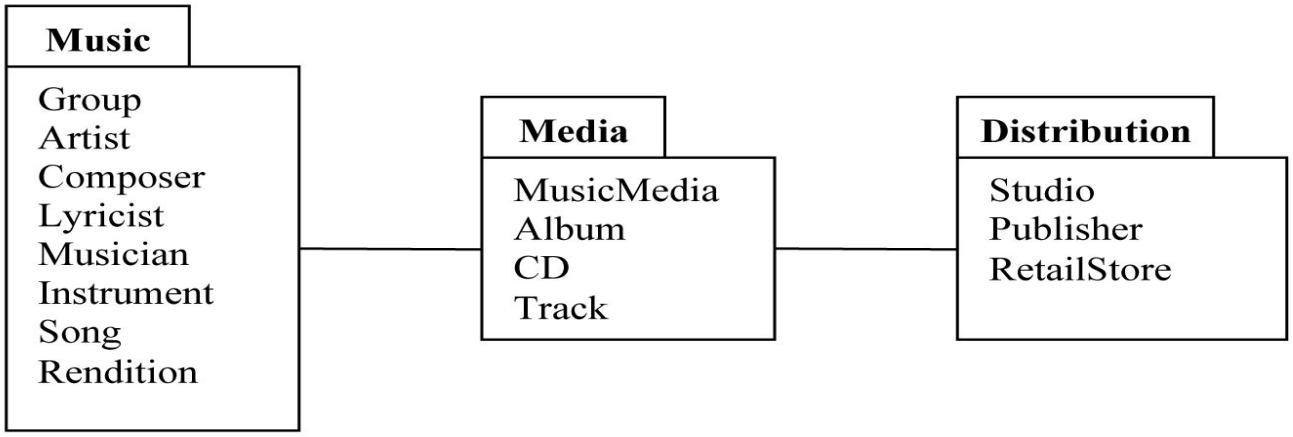
Örnek 12: *bilgisayar satis VT*



Uygun nitelikler belirleyebilirsiniz...

Örnek 13: Büyük VT tasarım örneği





örnek 13 (devam)