Veritabanı Yönetim Sistemleri (335)

Yrd. Doç. Dr. Ahmet Arif AYDIN

Özet

- Veri ile bilgi kavramları
- Veritabanı
- VTYS'nin kullanım alanları ve VTYS kullanmanın avantajları
- Dosya sistemleri ve problemleri
- VTYS kullanıcıları
- Günümüzde kullanılan VTYS'nin veri modelleri
- VTYS'lerinin kuramsal seviyeleri

Veritabanı Tasarım Aşamaları

1- **Gereksinim Analizi**
(*Requirement Analysis*)

- Kullanıcı isteklerinin ***toplantılarla*** belirlendiği aşamadır.
- Kullanıcıların Veritabanından beklentileri belirlenir.
- Veritabanına kaydedilecek veri tanımlanır.
- Veriyi işleyecek *uygulamalardan beklenilen özellikler* belirlenir.
- Öncelik ve performans gerektiren işlemler belirlenir.

Veritabanı Tasarım Aşamaları

1- **Gereksinim Analizi** (*Requirement Analysis*)

2- **Kavramsal Veritabanı Tasarımı** (*Conceptual Database Design*)

- İlk aşamada belirlenen istekler doğrultusunda
  - veritabanına kaydedilecek verinin ER (Entity-relationship) modeli oluşturulur.
- Veritabanına kaydedilecek verinin kullanıcılar ve geliştiriciler tarafından *ortak kullanılabilen* ve *yüksek-seviyeli görüntüsü oluşturulur.*
- ER Modeli *teknik bilgi sahibi olmayan* kullanıcıların verinin nasıl kaydedileceğini anlamasına ve tasarım sürecine dahil edilmesine yardımcı olur.

Veritabanı Tasarım Aşamaları

1- **Gereksinim Analizi**
(*Requirement Analysis*)

2- **Kavramsal Veritabanı Tasarımı**
(*Conceptual Database Design*)

3- **Mantıksal Veritabanı Tasarımı**
(Logical Database Design)

- İkinci aşamada belirlenen ER modelinin ilişkisel veritabanı şemasına dönüştürülür.

Veritabanı Tasarım Aşamaları

1- **Gereksinim Analizi**
(*Requirement Analysis*)

2- **Kavramsal Veritabanı Tasarımı**
(*Conceptual Database Design*)

3- **Mantıksal Veritabanı Tasarımı**
(Logical Database Design)

4- **Şema Yenilenmesi**
(Schema Refinement)

- 3. aşamada belirlenen şemanın problemleri ortaya çıkarılır ve tekrar düzenlenir.

Veritabanı Tasarım Aşamaları

1- **Gereksinim Analizi**
(*Requirement Analysis*)

2- **Kavramsal Veritabanı Tasarımı**
(*Conceptual Database Design*)

3- **Mantıksal Veritabanı Tasarımı**
(Logical Database Design)

4- **Şema Yenilenmesi**
(Schema Refinement)

5- **Fiziksel Veritabanı Tasarımı**
(*Physical DB Design*)

- İstenilen performans kriterlerinin kontrolü yapılır.
- **index tabloları** ve yeni **Tablolar** oluşturulur.

Veritabanı Tasarım Aşamaları

1- **Gereksinim Analizi** (*Requirement Analysis*)

2- **Kavramsal Veritabanı Tasarımı** (*Conceptual Database Design*)

3- **Mantıksal Veritabanı Tasarımı** (Logical Database Design)

4- **Şema Yenilenmesi** (Schema Refinement)

5- **Fiziksel Veritabanı Tasarımı** (*Physical DB Design*)

6- **Uygulama ve Güvenlik Tasarımı** (*Application and Security Design*)

- Uygulama ve güvenlik işlemlerinin tanımlandığı aşamadır.
- UML diyagramları yardımıyla uygulamaların ve tanımlanan görevlerin işlem aşamaları belirlenir.
- Her bir varlığın iş akışı şemasında görev ve etki alanı tanımlanır.
- Veritabanına erişilebilen ve erişimi sınırlı olan kısımları tanımlanır.

Entity-Relationship Model

Varlık-iliski modeli

- **Varlık** (entity) ve **varliklar arasındaki ilişkiyi** (relationship) **görsel olarak** diyagramlar yardımıyla tanımlamayı sağlar.
- Yaygın olarak kullanılan bir ***anlamsal veri modeli*** (semantic data model) dir.

ER Model:Varlık-İlişki Modeli

Bir veritabanının tasarım aşamasında

veriyi modellemek için kullanılacak nesneleri ve

nesneler arasındaki ilişkileri tanımlamak için kullanılır.

ER modelini oluşturmak kullanıcıların istedikleri bir veritabanını oluşturma sürecinde ki **ilk somut aşamadır.**

ER Model: Entity

Gerçek hayatta bulunan ve **diğer nesnelerden ayırt edilebilen** nesnelere **varlık** (entity) denir.

- Üniversitemizde ki her bir öğrenci
- Bölüm başkanımız
- Bölümümüzde bulunan her bir Öğretim üyesi
- Ders

ER Model: Entity Set

Benzer nesnelerin *oluşturduğu kolleksiyona* **varlık kümesi** (entity set) denir

- Öğretim üyeleri
- Öğrenciler
- İdari personel

ER Model: Entity Set

Varlık kumesi (entity set) ayrık olmayabilir

- Öğrenciler kümesinde 1. ve 4. sınıftan öğrenciler bulunabilir.
- İnönü üniversitesi bünyesinde bulunan öğretim üyeleri farklı bölümlerde olmalarına rağmen Akademik Personel varlık kümesi içerisinde değerlendirilebilir

ER Model: relationship

- **Varliklar arasinda olan iliski** (relationship) olarak tanımlanır.
  - öğrenciler ve dersler arasında **dersi alır** ilişkisi
  - öğretim üyesi ve dersler arasında okutur ilişkisi
  - İdari personel ile bölüm arasında çalısır ilişkisi
  - Rektör ve üniversıte arsında yönetir ilişkisi

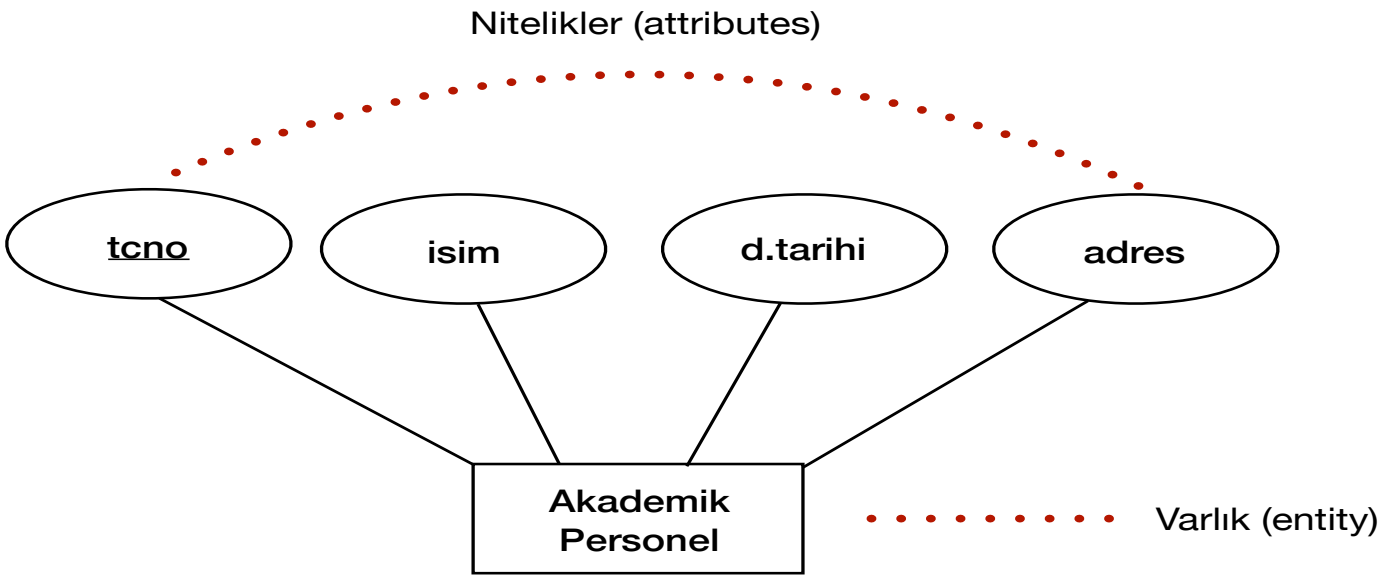
ER Model: Attributes

Her bir **varlık** (entity) **nitelikler** (attributes) ile tanımlanır

- Bir varlık kümesi içinde bulunan nesnelerin benzer nitelikleri bulunmaktadır
- Niteliklerin tanımlamaları yapılmalıdır.
  - öğrenci nosu (Int)
  - adı soyadı (30 karakterden oluşan String)
  - TC No (Int)

ER Model:Attributes

- Herbir varlık kümesi için bir **anahtar (key)** seçilmelidir
- **Birden fazla** anahtar adayı varsa **bir tanesi primary key** olarak seçilir



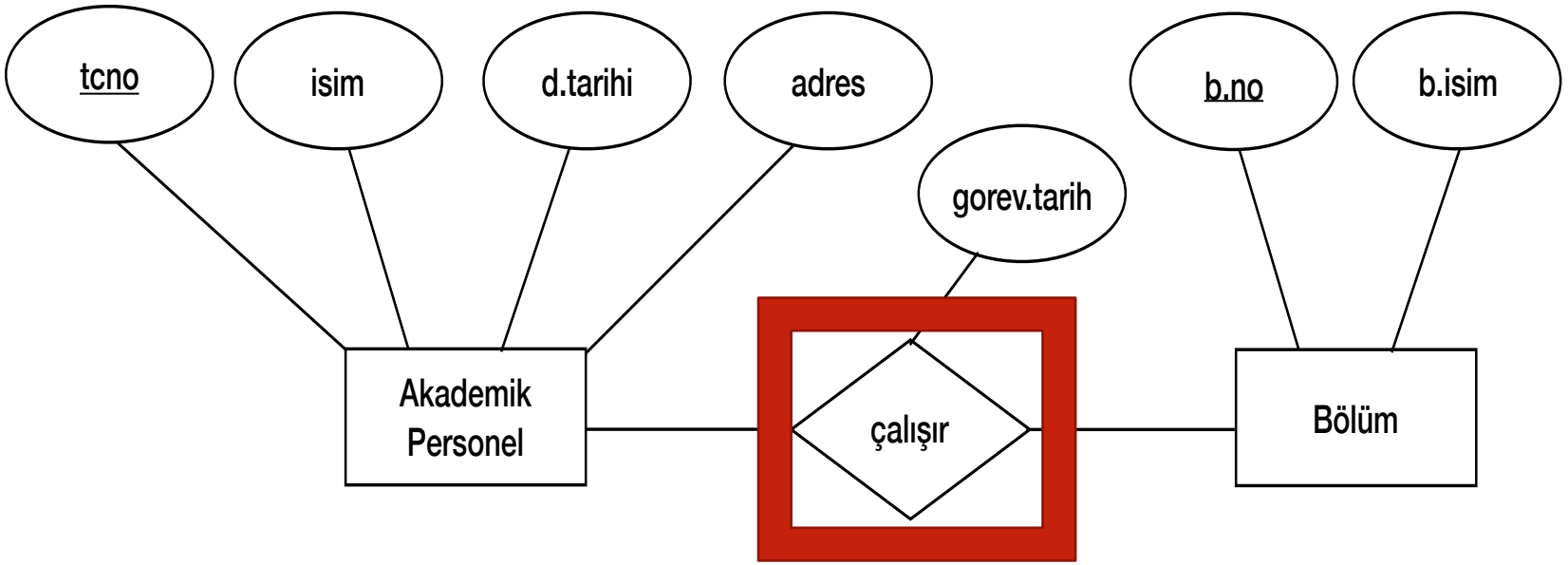
Akademik Personel varlık seti

ER Model: Relationship set

- İki yada data fazla entity'nin bağlantılı olmasına **ilişki** (*relationship*) denir
- Benzer ilişkiler **ilişki kümesi** (relationship set) içerisinde tanımlanır
- Akademik personelin her biri bir varlik seti (E_i) içinde tanımlanır

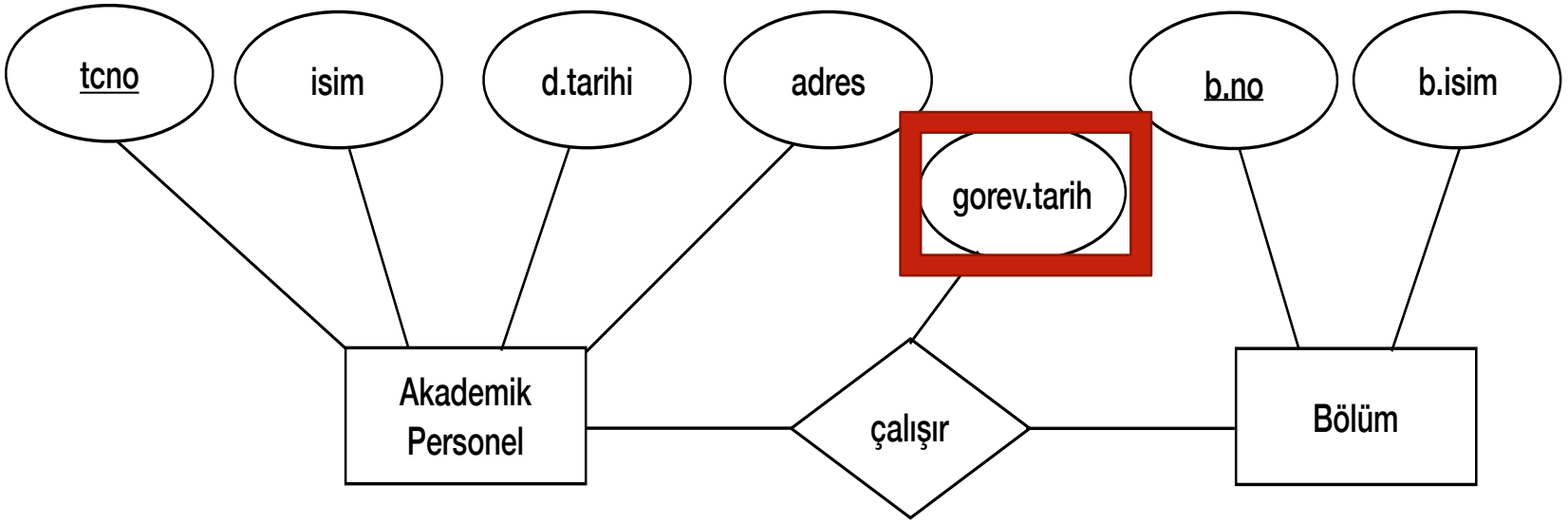
$$\{ (ap_1, ap_{2,\ldots,} ap_n) \mid ap_1 \in E_1 , ap_2 \in E_2 , \ldots, \mid ap_n \in E_n \}$$

ER Model: Relationship set



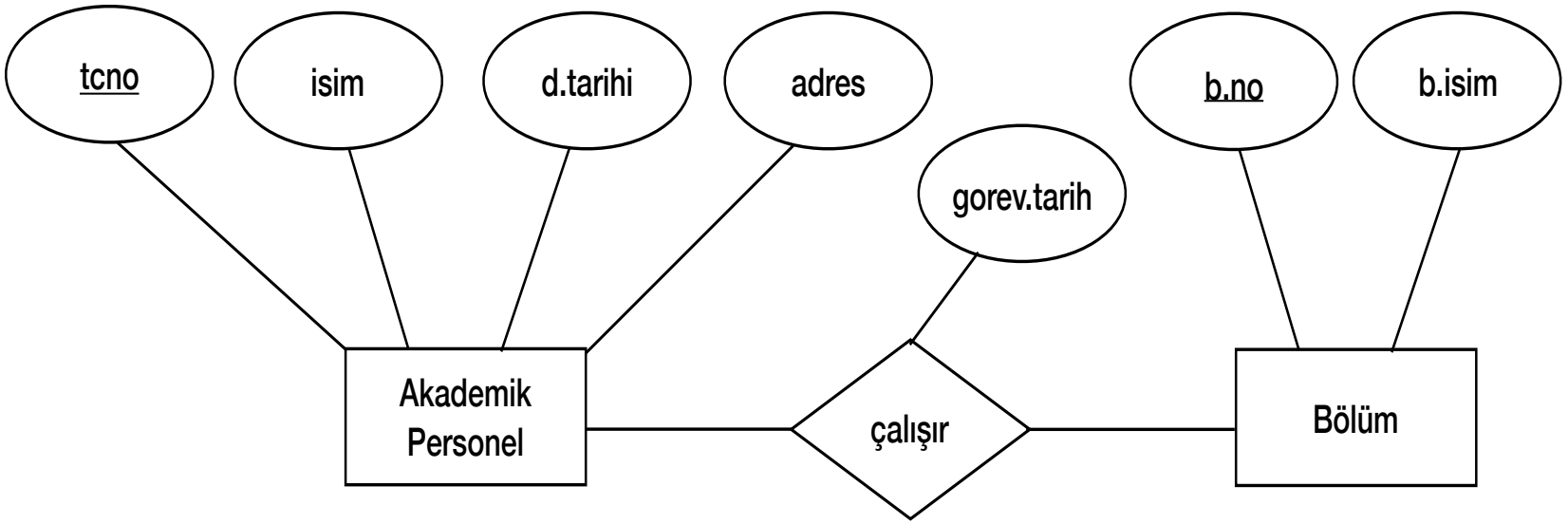
- Akademik Personel varlık kümesinin Bölüm varlık kümesi arasında **çalışır ilişkisi** tanımlanabilir.

ER Model: Descriptive Attributes



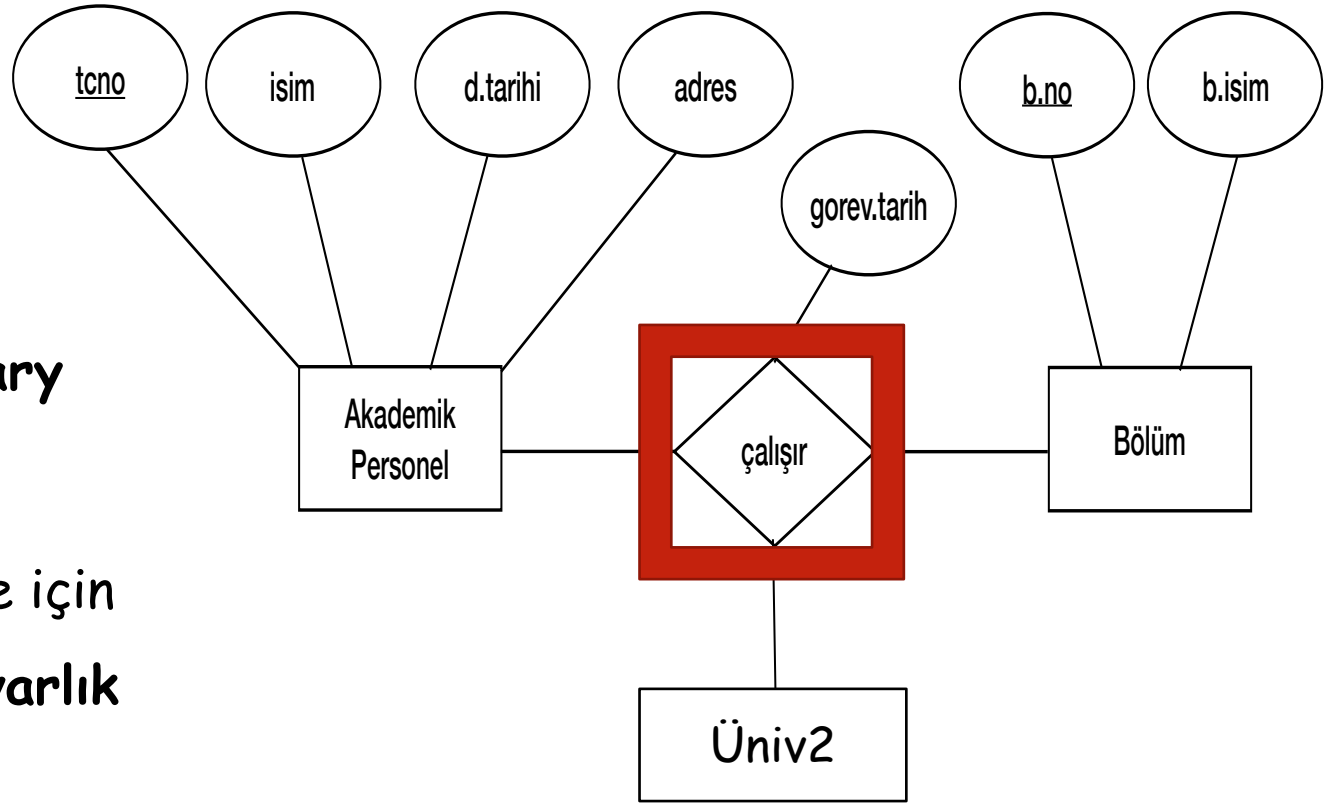
- **ilişkilerin açıklayıcı nitelikleri** (descriptive attributes) bulunmaktadır
  - çalışır ilişkisinin **gorev.tarih** niteliği bulunmaktadır.

ER Model: Relationship



- Varlıklar arasındaki ilişki tek (**unique**) olarak tanımlanmalıdır.
- Akademik personel ve bölüm varlıklarının ilişkisi içerisinde bulunan gorev.tarih niteliği unique'dir birden fazla olamaz.

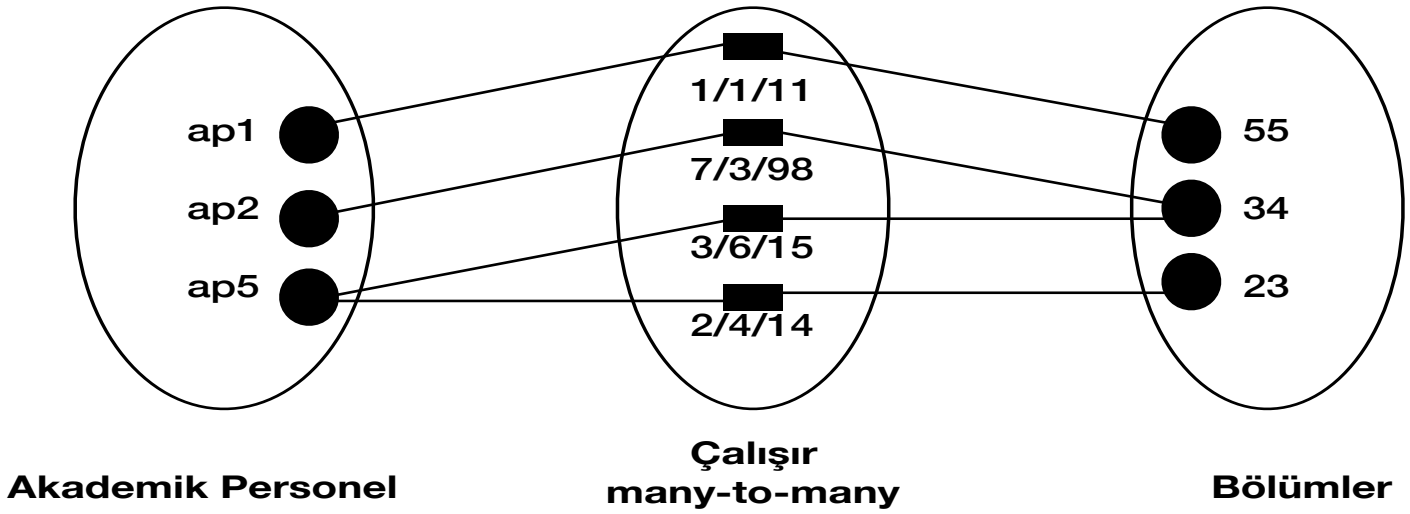
ER Model:Ternary Relationship



- Üç varlık arasındaki ilişki üçlü ilişki **ternary relationship** olarak tanımlanır.
- ap1 ögretim görevlisi başka bir üniversite için de çalışıyorsa çalışır ilişki kümesine **bir varlık eklenebilir.**

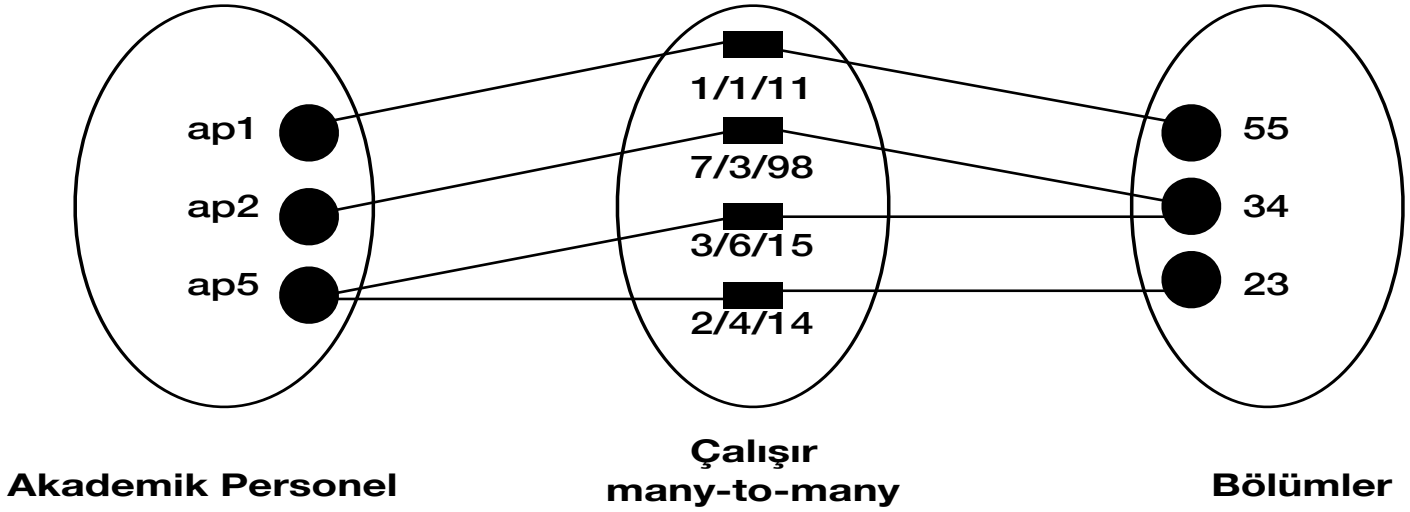
ER Model: instance

İlişki kümesinin bir örneği (*instance*) de bir ilişki kümesidir.



ER Model: Many-to-many

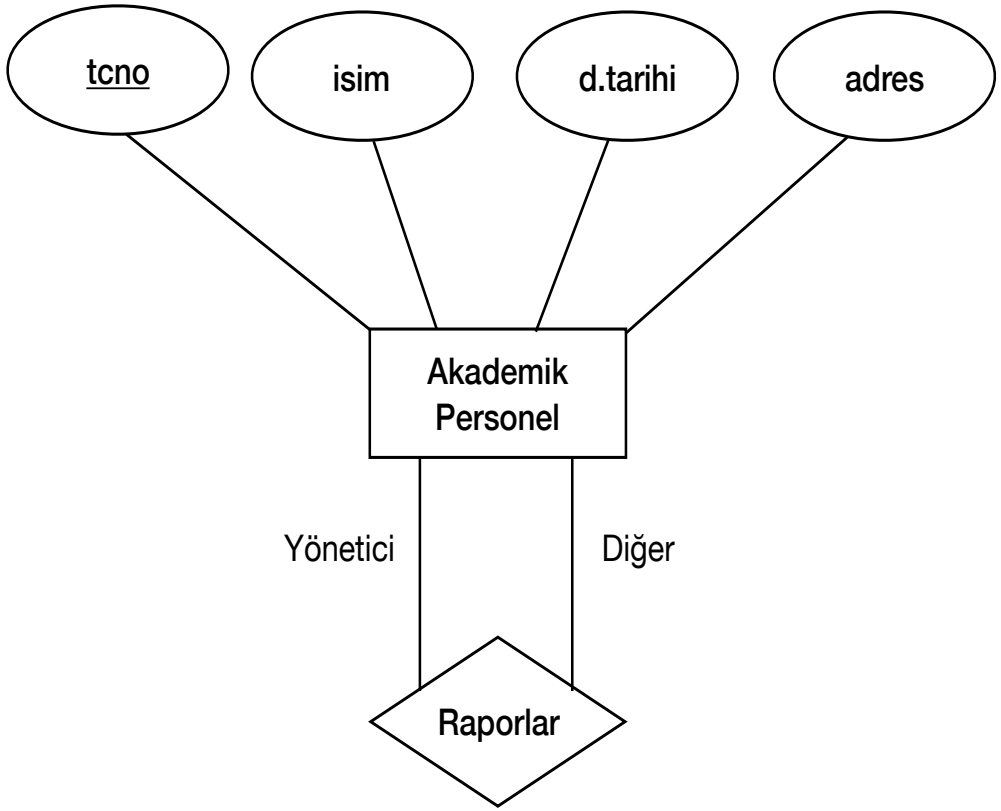
İlişki kümesinin bir örneği (*instance*) de bir ilişki kümesidir.



- Çalışır ilişki kümesine göre bir *akademik personel birden fazla bölümde* çalışabilir (ap5) .
- Bir bölümde birden fazla akademik personel çalışabilir. (34 nolu bölüm)
- Akademik personel ve bölümler varlık kümeleri arasında **many-to-many** ilişkisi bulunur.

ER Model: Relationship

- Bir varlık kümesinde ilişki sorumluluğu farklı olan varlıklar varsa sorumluluk tanımları ilişkide belirtilir.
  - Yönetici ve diğer (yönetilen) personel ile raporlar arasında bulunan farklı ilişki seviyesi belirlenebilir.



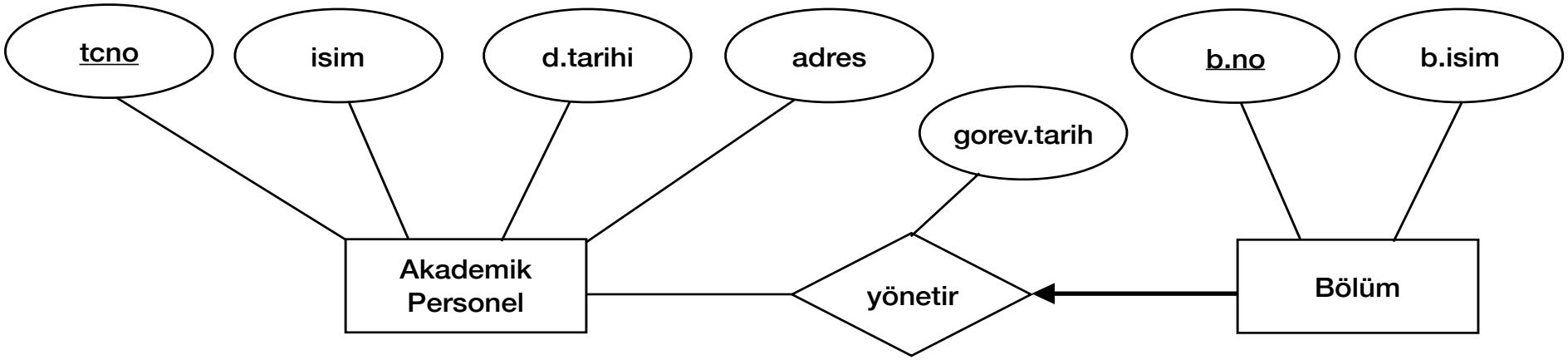
ER Model: key constraints

ER modelinde

bazı semboller (ok, kalın cizgi) kullanılarak

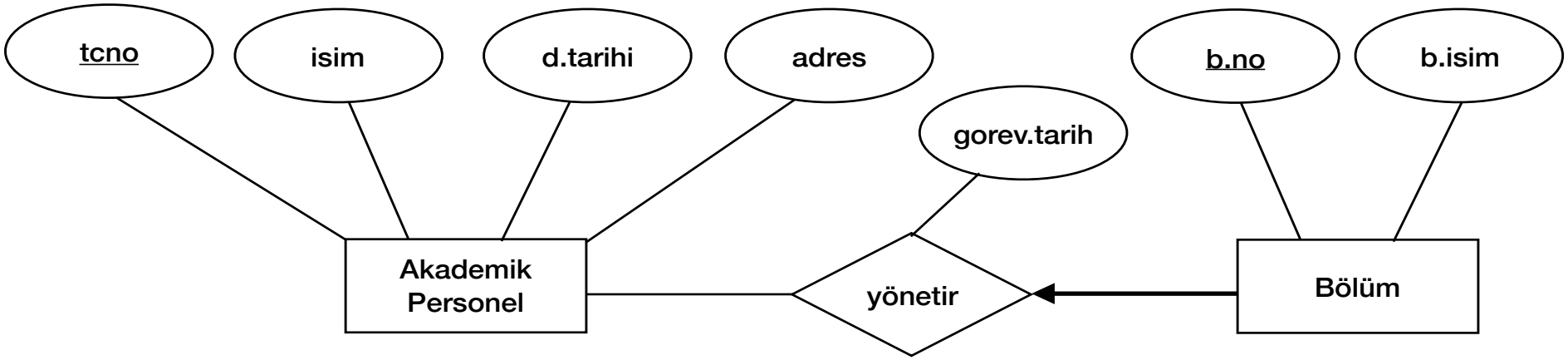
anahtar kısıtlamalar (key constraints) tanımlanmaktadır.

ER Model: one-to-many



- Bir akademik personel birdan fazla bölümü yönetebilir fakat ***aynı konum birden fazla kişi tarafından paylaşılamaz***.
  - Bu kısıtlama **ok ile ifade edilmiştir**
  - bu ilişki **one-to-many** olarak tanımlanmaktadır.

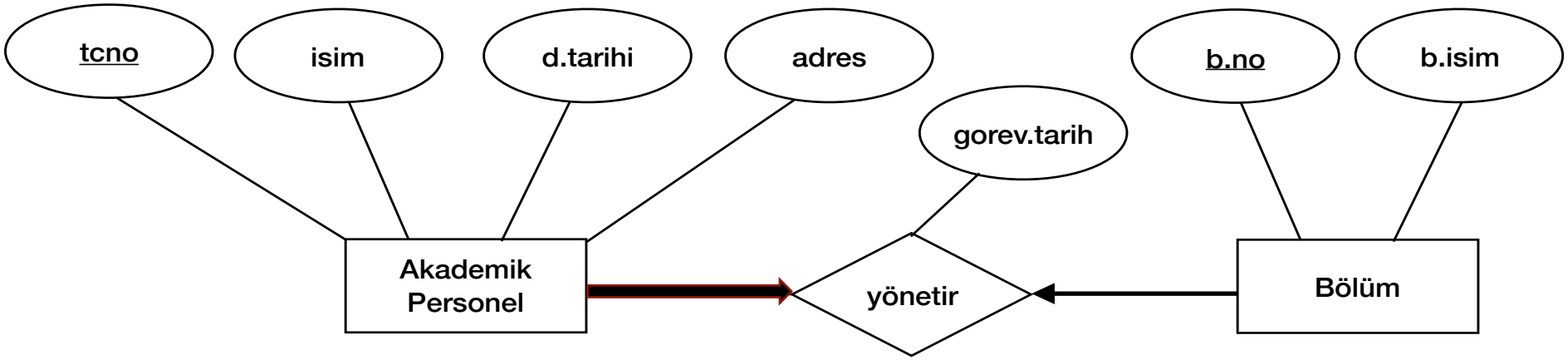
ER Model: one-to-many



- Mühendislik fakültesinin **sadece bir dekanı olur**

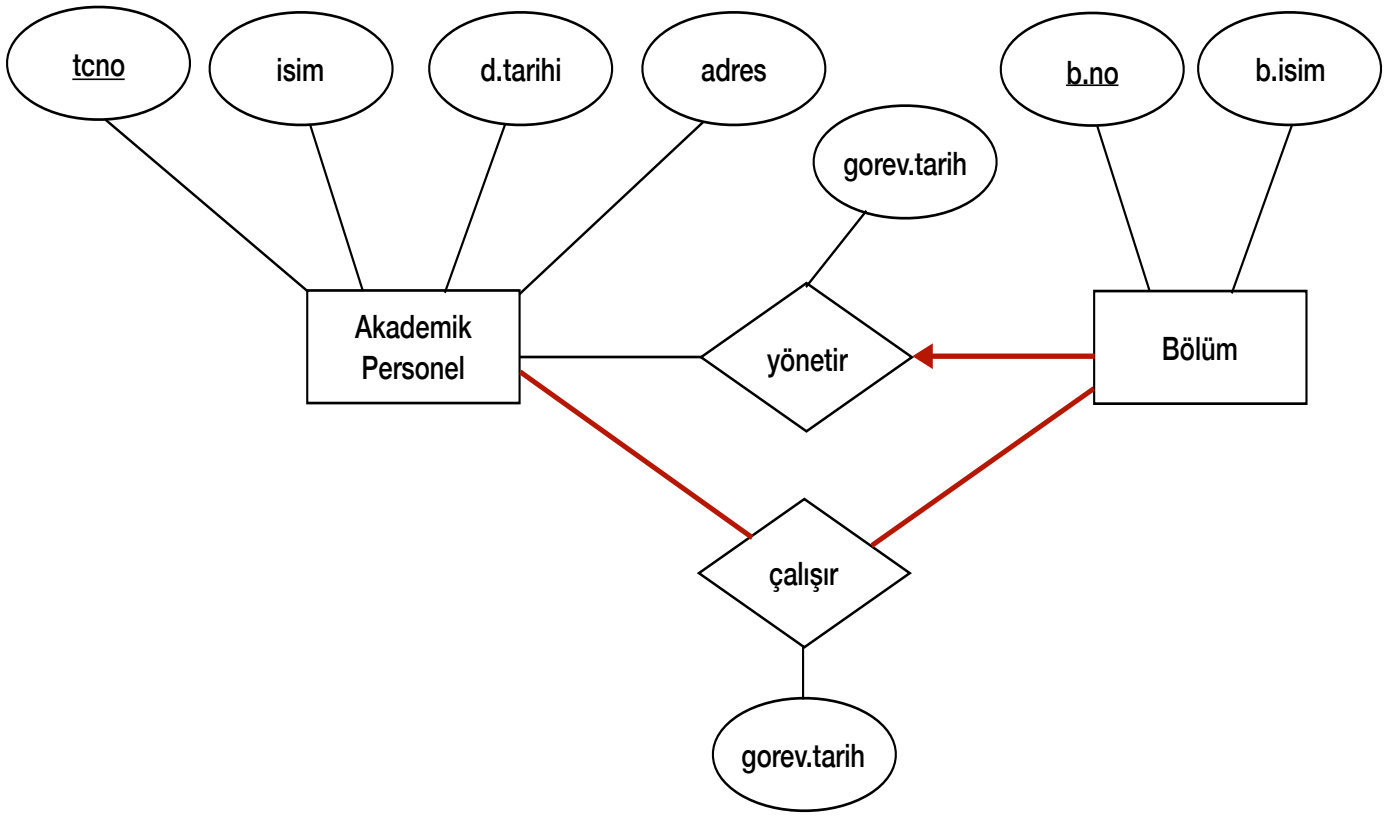
  fakat aynı kişi baska bir fakültede de dekanlık yapabilir.

ER Model: one-to-one



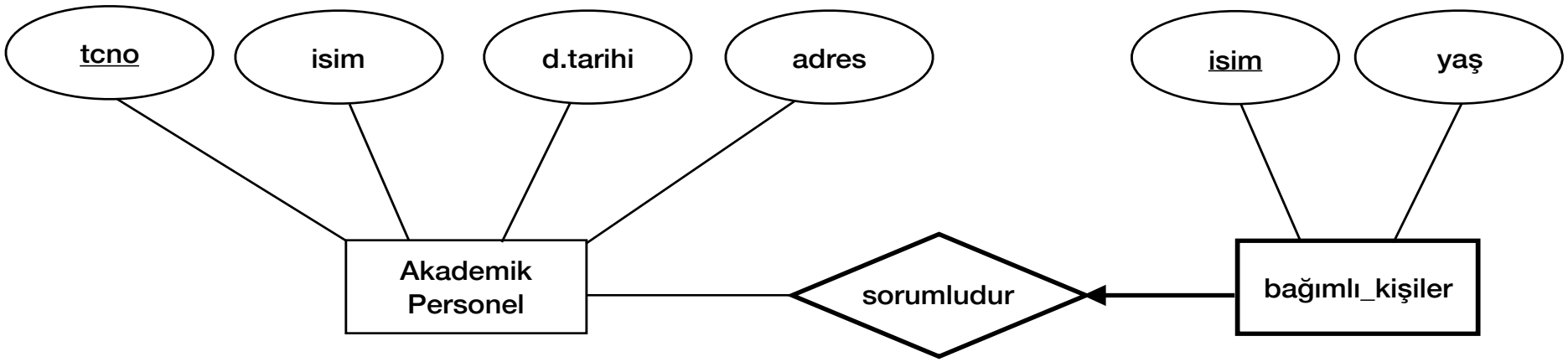
- Birden fazla fakültede dekan olma şartı kısıtlanabilir
- Sadece bir akademik personelin bir bölümü yönetmesi isteniyorsa **(one-to-one)**
- Akademik personelden yönetir ilişkisine dogru bir ok ile bu kısıtlama eklenebilir.

ER Model: total participation



Akademik personelin hepsi çalışır ilişkisinde yer aldığından akademik personel ile çalışır arasında kalın bir çizgi kullanılır ve **toplam kalıtım** *(total participation)* olarak adlandırılır.

ER Model: weak entities



- Bir varlık kümesine **geçici olabilen** varlıklar eklenebilir ve zayıf varlıklar (weak entities) olarak isimlendirilir.
- akademik personel varlık kümesine **sorumludur** ilişkisi ile birlikte **bakmakla yükümlü oldugu kişiler** eklenebilir.

Sınıf Hiyerarşileri (Class Hierarchies)

- Varlık kümelerinin kendi içerisinde alt sınıflara ayrılmaları dogaldır.
- Akademik personelin kendi içerisinde sınıflandırılması gerekmektedir
- Akademik personel içerisinde
  - Uzman , Okutman
  - Araştırma Görevlisi
  - Yardımcı Doçent, Doçent
  - Professor

Class Hierarchies: Inheritance

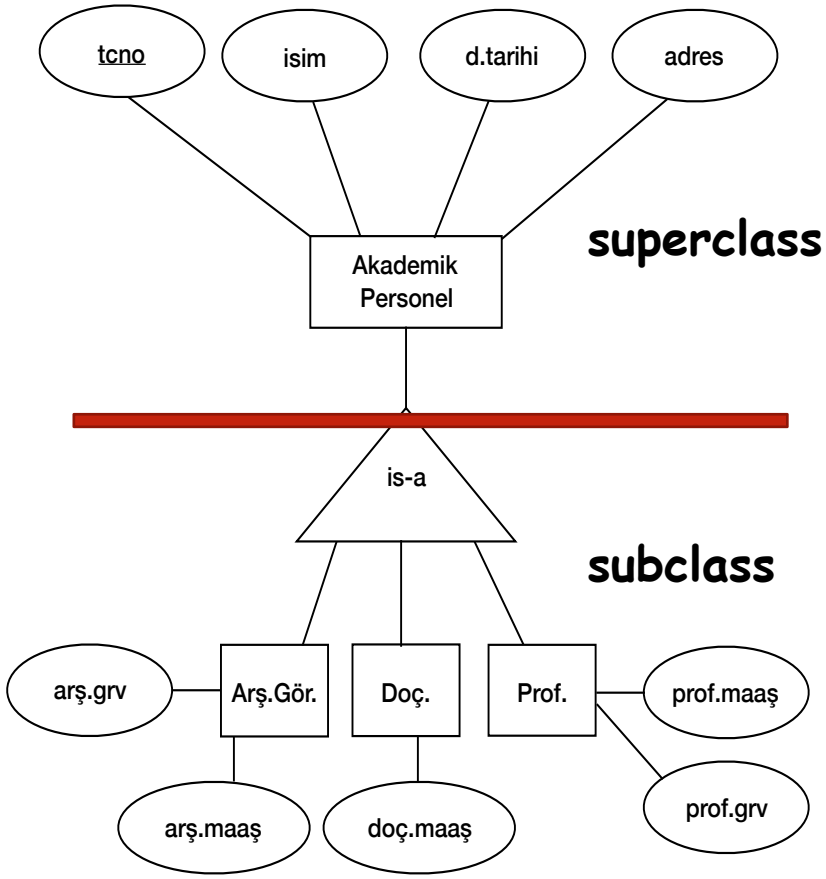
Her bir Akademik personel Akademik Personel varlık kümesinde **tanımlı olan** *tüm nitelikleri* **kalıtsal olarak devralır** (***inheritance***)

Akademik Personel varlık kümesinde **tanımlı olmayan** başka nitelikler ise **alt sınıf (subclass)** oluşturularak eklenebilir.

Class Hierarchies:superclass,subclass

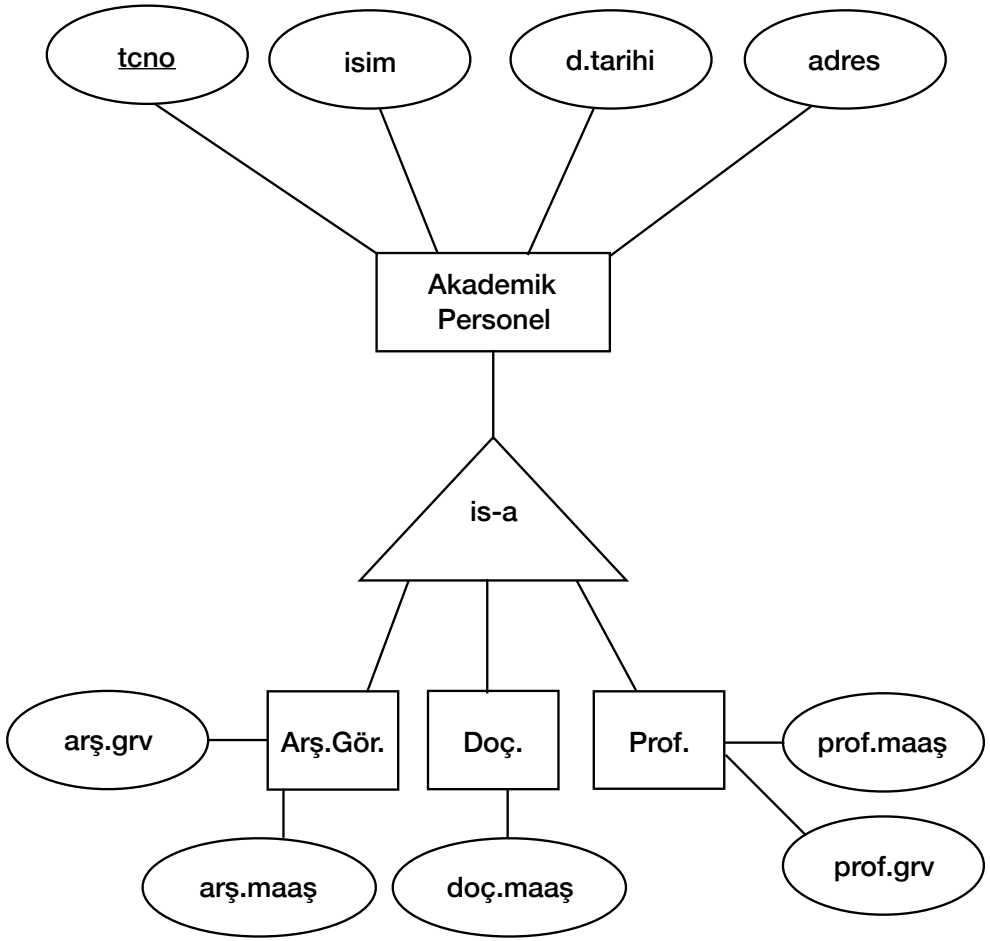
Hiyerarşik sınıf tanımlamalarında

1. **üst sınıf** (superclass) tanımlanır
2. **üst sınıfı** tanımlayan **nitelikler** belirlenir
3. **alt sınıf** (subclass) lar tanımlanır
4. alt sınıfa ait nitelikler belirlenir



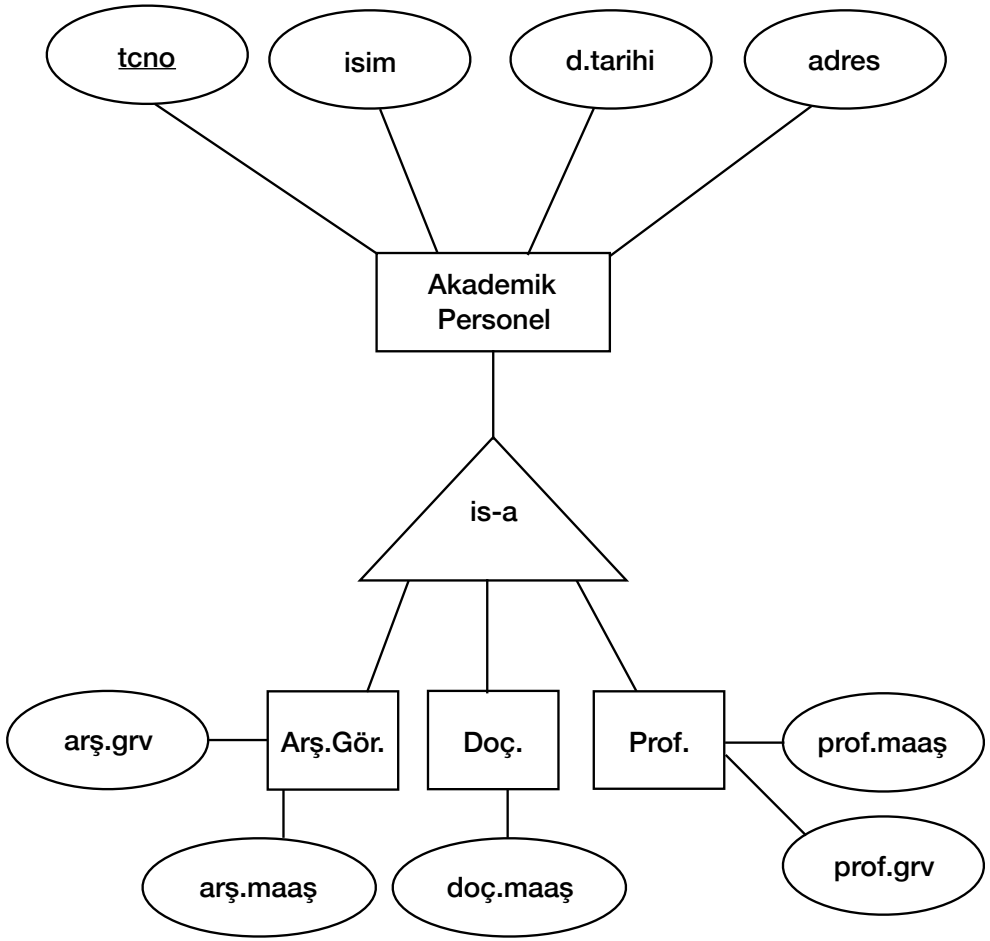
Class Hierarchies: is-a ilişkisi

- Alt sınıfta (subclass) bulunan varlıgın bir ust seviyedeki (superclass) varlığın alt sınıfdaki bir üyesi olduğunu belirtir.
- Alt sınıfta olan bir varlık kumesinin üst sınıf ile olan alakası **is-a ilişkisi** ile tanımlanır



Class Hierarchies: is-a ilişkisi

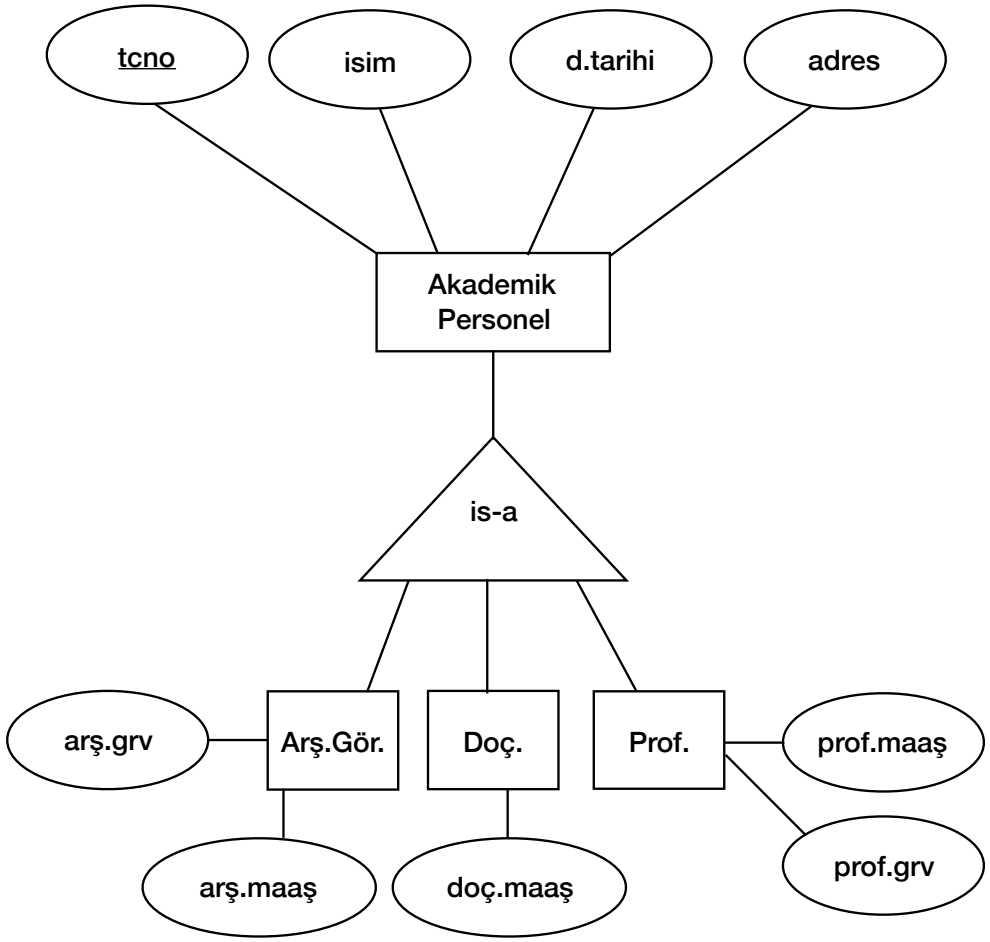
- "araştırma görevlisi **bir** akademik personel **dir**"
- "Doçent **bir** akademik personel **dir**"
- "Prof **bir** akademik personel **dir**"



Class Hierarchies

Bir Prof.

- akademik personel varlık kümesinin bütün niteliklerini içerir
- Kendinine ait özel nitelikleri bulunmaktadır
- Diğer akademik personelden farklı olan görev tanımı ve sorumlulukları bulunmaktadır.



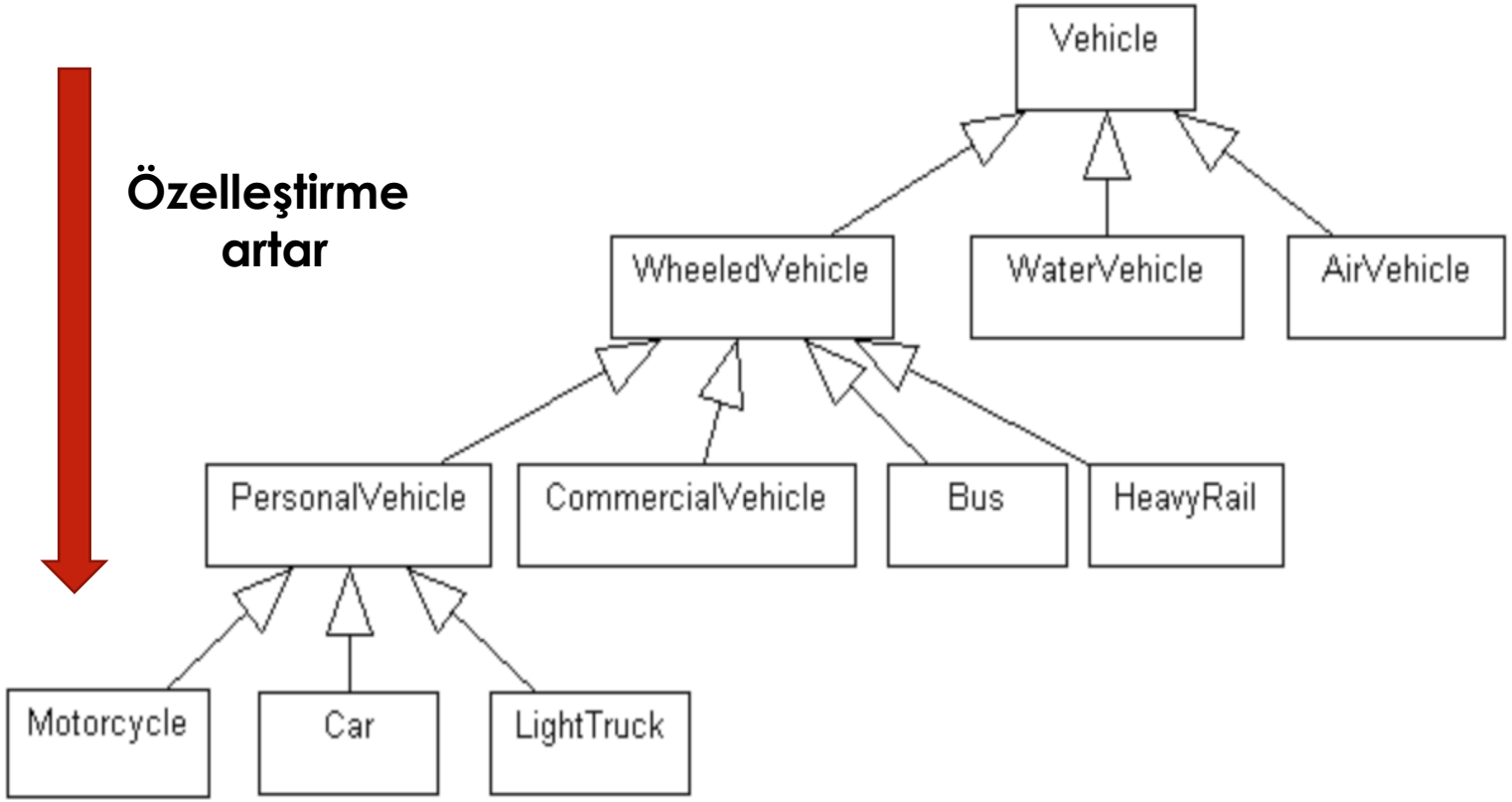
Class Hierarchies

Sınıf hiyerarşik yapısı

özelleştirme (specialization) ve

genelleştirmeyi (generalization) gerceklestirmeyi sağlar

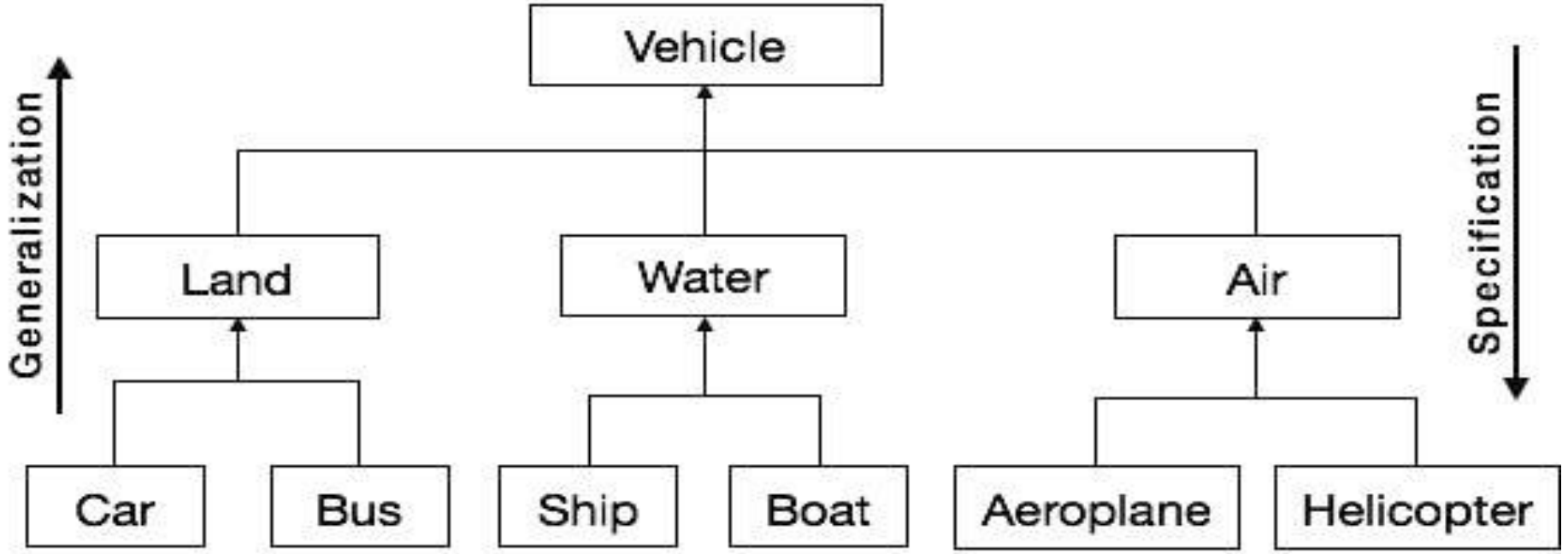
Class Hierarchies: specialization



Özelleştirme sayesinde üst sınıftan devralınan niteliklerden faklı olan ve sadece alt sınıfta bulunan varlıklara ait niteliklerin tanımlanmasini saglar.

https://users.cs.jmu.edu/bernstdh/web/common/lectures/slides_inheritance-introduction.php

Class Hierarchies: generalization



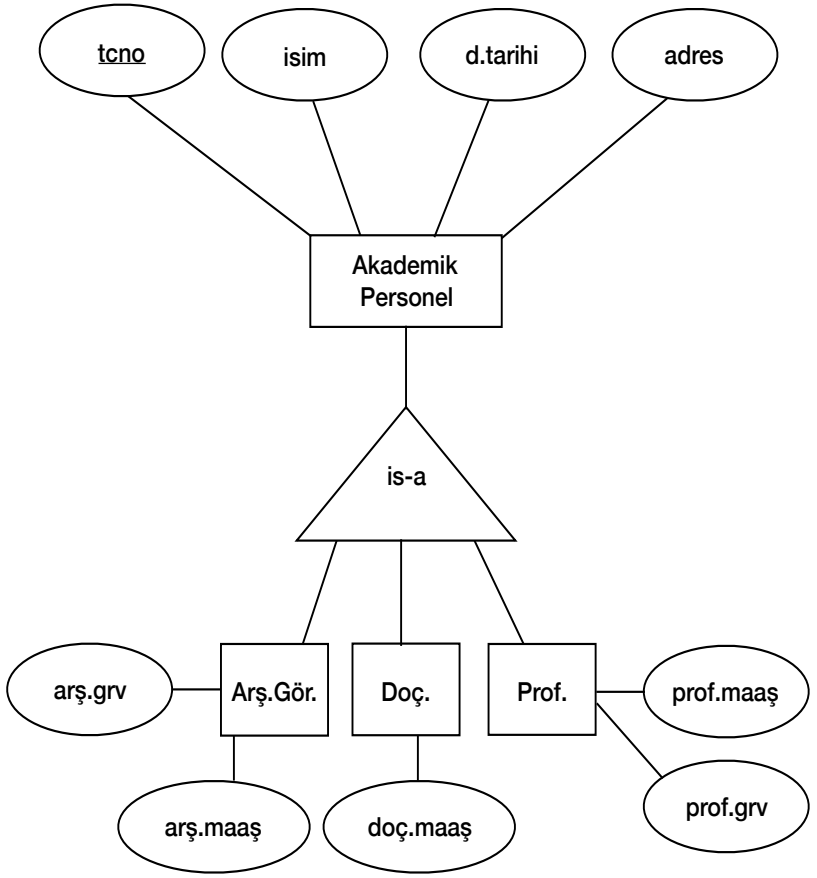
Genelleştirme ortak özellikleri bulunan varlıkların bir arada toplanabilmesini sağlar.

https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_object_oriented_model.htm

Is-a kısıtlamaları: overlap

Çakişma (*overlap*) kısıtlaması

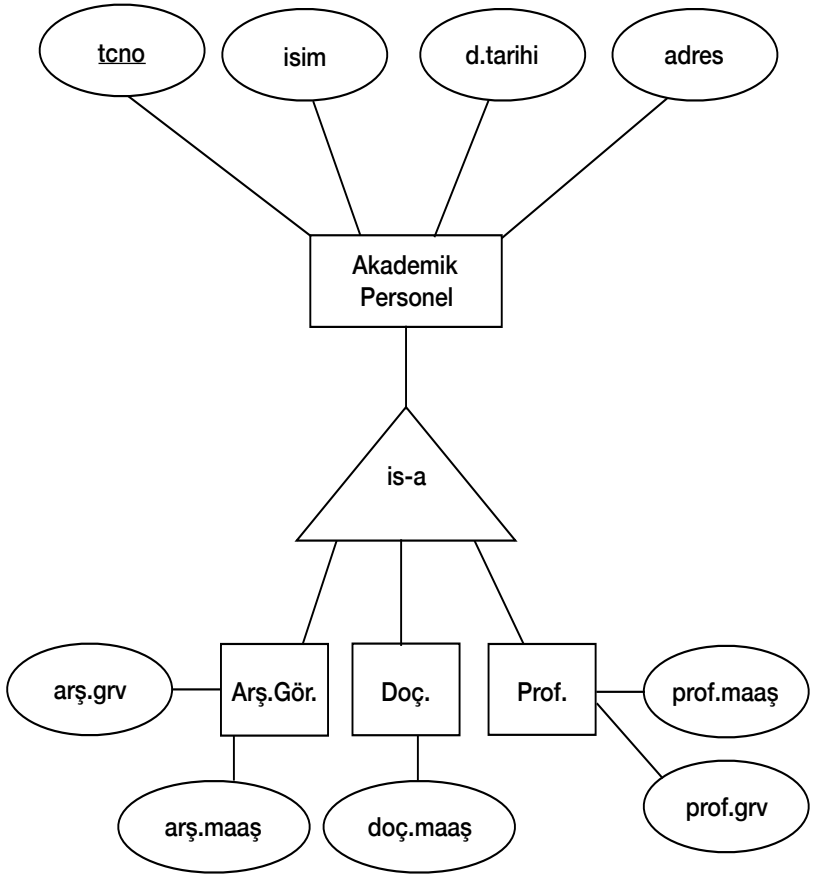
- iki alt sınıfın aynı varlığı içermesini engeller.
- Bir akademik personel aynı anda hem Doç hemde Arş.Gör. olamaz



is-a kısıtlamaları: covering

Kapsama (*covering*) kısıtlaması

- alt sınıftlarda bulunan varlıkların toplamı üst sınıfı vermesi prensibidir.
- Arş.Gör. , Doç. ve Prof. KAPSAR Akademik Personel.



Aggregation (Kümeleme)

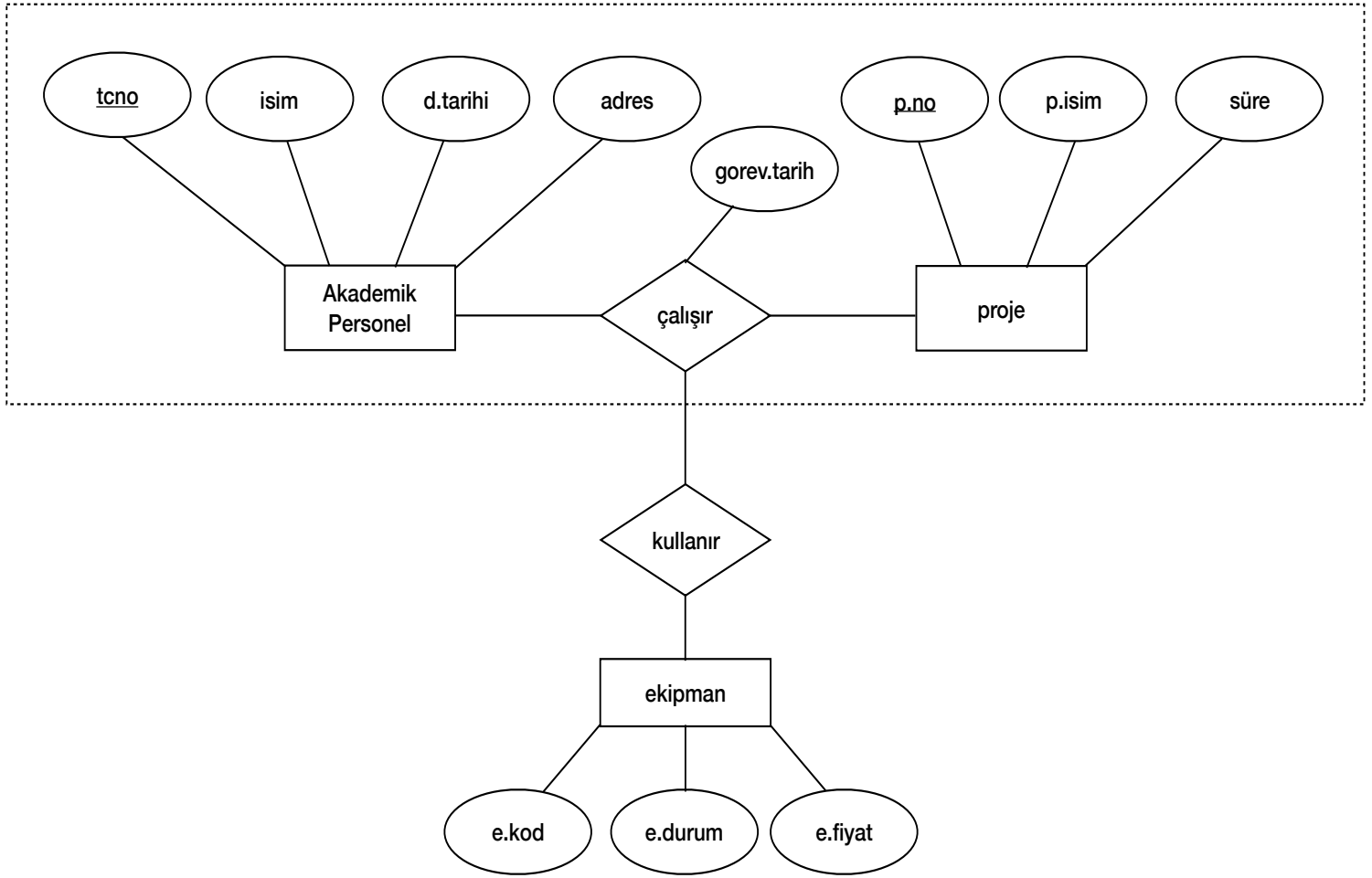
Kümeleme (Aggregation)

- Her bir ilişki seti kendi başına anlam ifade etmektedir
- Birden fazla ilişki setinin toplanarak başka bir ilişki setine bağlanmasına **kümeleme** denir.

https://ozlemerden.wordpress.com/2012/02/12/genelleme-kumeleme-ve-iliskisel-model/

Aggregation (Kümeleme)

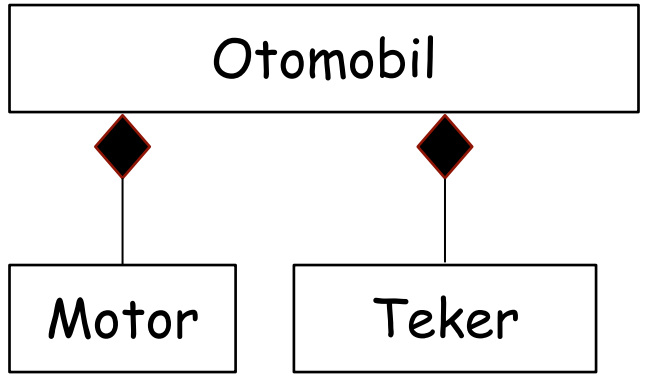
- Akademik personel varlık kümesinin proje ile olan ilişkisi çalışır ilişkisi ile tanımlanmıştır.
- Çalışır ilişkisi kümeleme yardımıyla kullanır ilişkisine bağlanmıştır.
- Projede kullanılan ekipmanlar ise kullanır ilişkisine bağlanmıştır.



https://ozlemerden.wordpress.com/2012/02/12/genelleme-kumeleme-ve-iliskisel-model/

Kompozisyon (Composition)

- Bir varlığı oluşturan alt sınıfta ki varlıklar ile ilişkisi kompozisyon olarak tanımlanır.
- Kompozisyon ilişkisi ile baglı olan alt sınıflar ust sınıf dan ayrılmazlar.



ER Model ile Kavramsal Tasarim

Bir ER Modelinin tasarım asamasında aşagıdaki sorular ortaya cıkabilir:

- Bir kavram **varlık** (entity) olarak mı yoksa **nitelik** (attribute) olarak mı tanımlanmalı?
- Bir kavram **varlık** (entity) olarak mı yoksa **ilişki** (relation) olarak mı tanımlanmalı?
- ER modelinde bulunacak olan varlık ve ilişki setleri hangileridir?

ER Model ile Kavramsal Tasarim

Belirlenen ihtiyaclar cervevesinde tasarim yapilir

- Bir varlık kümesinin nitelikleri belirlenirken ihtiyaca göre bir nitelik varlık kümesi olarak tanımlanabilir.
- bir akademik personelin ikametgah adresi için tek bir nitelik yetebilirken
  - detaylı aramaların yaptırılabilmesi için **adress varlık kümesi** oluşturulabilir
  - posta kodu, şehir, mahalle ve başka istenilen nitelikler eklenebilir
  - **adress varlık kümesi** akademik personel varlık kümesi ile ilişkilendirilir.

ER Model ile Kavramsal Tasarim

- **Farklı kişiler** ER modelini kullanılarak
  - aynı problemin çözümü için farklı tasarımlar sunabilirler.
- Arzu edilen ER model tasarımında
  - veritabanına kaydedilecek **bilgilerin tekrarlanmaması**
  - Veriye zamanında hızlı bir biçimde erişim
  - Sorguların makul zaman içerisinde tamamlanması

ER Model ile Kavramsal Tasarim

- Büyük firmalarda yapılan tasarimlar birden fazla kişinin calışmasını (gelistirici-DB yöneticisi- Kullanicilar) gerektirir.
- Zaman içerisinde tespit edilen tasarım eksiklileri tekrarlanmali (iterative) olarak giderilir.
- Yüksek seviyeli tasarim farklı uzmanlık alanlarına sahip olan kişilerin anlayabileceği üst seviyede olmalidir ki her bir katilimci kendi bilgi birikimi sayesinde tasarıma katkı saglayabilsin.

UML: Unified Modeling Language

A picture is worth a thousand words !

- 'Bir resim binlerce kelime değerindedir'
- UML in önemini ve işlevselliğini açıklamaktadır
- UML yazılım sistemlerinden öte diğer sistemlerin tasarım aşamalarında da kullanılabilir.

UML: Unified Modeling Language

Bir yazılım sisteminin

tasarım ihtiyaclarını tanımlama aşamasından

teslimatına kadar gecen evrelerde yazılımı

tanımlamak, görselleştirmek ve *dokumantasyon* için

yaygın olarak kullanılan standart dil UML'dir.

UML: Unified Modeling Language

- 1997 yılında OMG (Object Management Group) tarafından geliştirilmiştir.
- UML **resimsel** olarak bir yazılımın taslağını ortaya koymaktadır.
- UML digramlarını program koduna cevrilebilmektedir.

https://www.tutorialspoint.com/uml/uml_overview.htm

UML: Kullanım Aşamaları

İş modelleme (Businnes Modeling):

- Tasarımı yapılacak olan yazılımın çözümü gerçekleştirecek iş modelinin etraflıca kavranmasını sağlamaktır.
- İş alanının özel ihtiyaçlarını tanımlamak için kullanılır.

Sistem Modelleme (System Modeling)

- Geliştirilecek olan yazılımın gereksinimlerinin belirlenmesi sürecini kapsar.

UML: Kullanım Aşamaları

Kuramsal Veri Modelleme (Conceptual Data Modeling)

- ER modelinin oluşturulduğu aşamadır.

Fiziksel Veritabani Tasarımı (Physical Database Modeling)

- Fiziksel olarak veritabanını oluşturmayı sağlar.

Donanım Sistem Modelleme (Hardware System Modeling)

- Uygulamada kullanılan donanımı tanımlamak için kullanılır.

UML: Nesne tabanlı programlama

Nesne tabanlı programlamada UML aktif bir biçimde kullanılmaktadır:

Yazılımı oluşturan nesneleri

ve nesneler arasındaki ilişkileri tanımlamayı sağlar

UML: Nesne tabanlı programlama kavramları

Nesne (Object)

- Programları oluşturur
- Sınıflardan oluşan varlıkları temsil eder

UML: Nesne tabanlı programlama kavramları

Sınıf (Class)

- Fonksiyonlar ve veriler yardımıyla oluşturulur
- İçerisinde alan (field) , method, yapıcılar (constructors) ve diğer özellikleri barındırır
- programların daha anlaşılır olmasını sağlarlar.

UML: Nesne tabanlı programlama kavramları

Soyutlama (Abstraction)

Bir sistemin arka planında olan **komplex yapısının gizlenip**

basit bir ara yüzle kullanıcının kullanımına sunulmasıdır.

Java: String Class

UML: Nesne tabanlı programlama kavramları

Kapsulleme (Encapsulation, information hiding)

- Bir birleriyle alakalı kodların beraber bir araya toplanır
- Kodlama detaylarının dışarı gösterilmez

UML: Nesne tabanlı programlama kavramları

Kalıtım (inheritance)

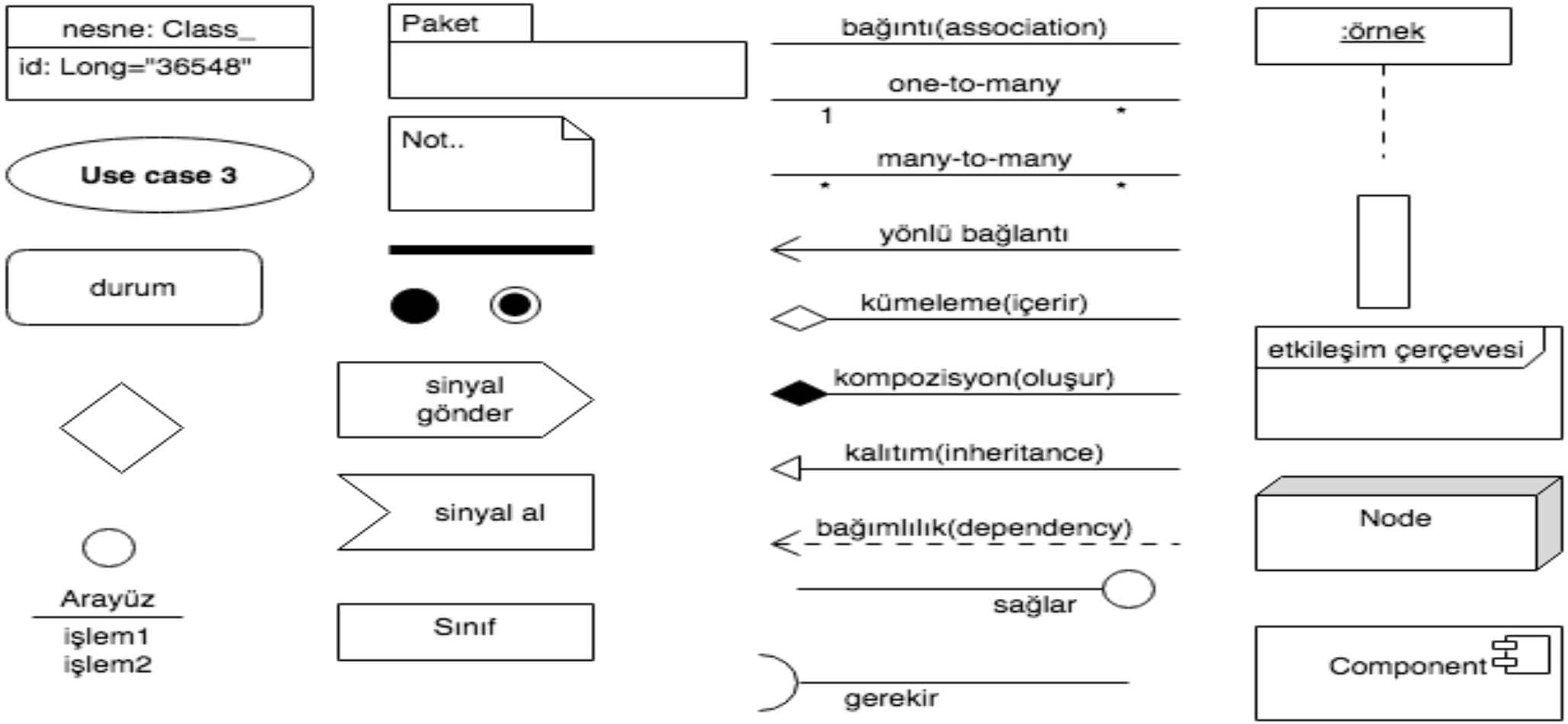
- Üst sınıfın özelliklerini katılsal olarak devralmaktır
- Kod tekrarını önler ve alt sınıflara yeni özellikler eklenmesine imkan sağlar

UML: Nesne tabanlı programlama kavramları

Polymorphism (çokbiçimlilik, çokçeşitlilik)

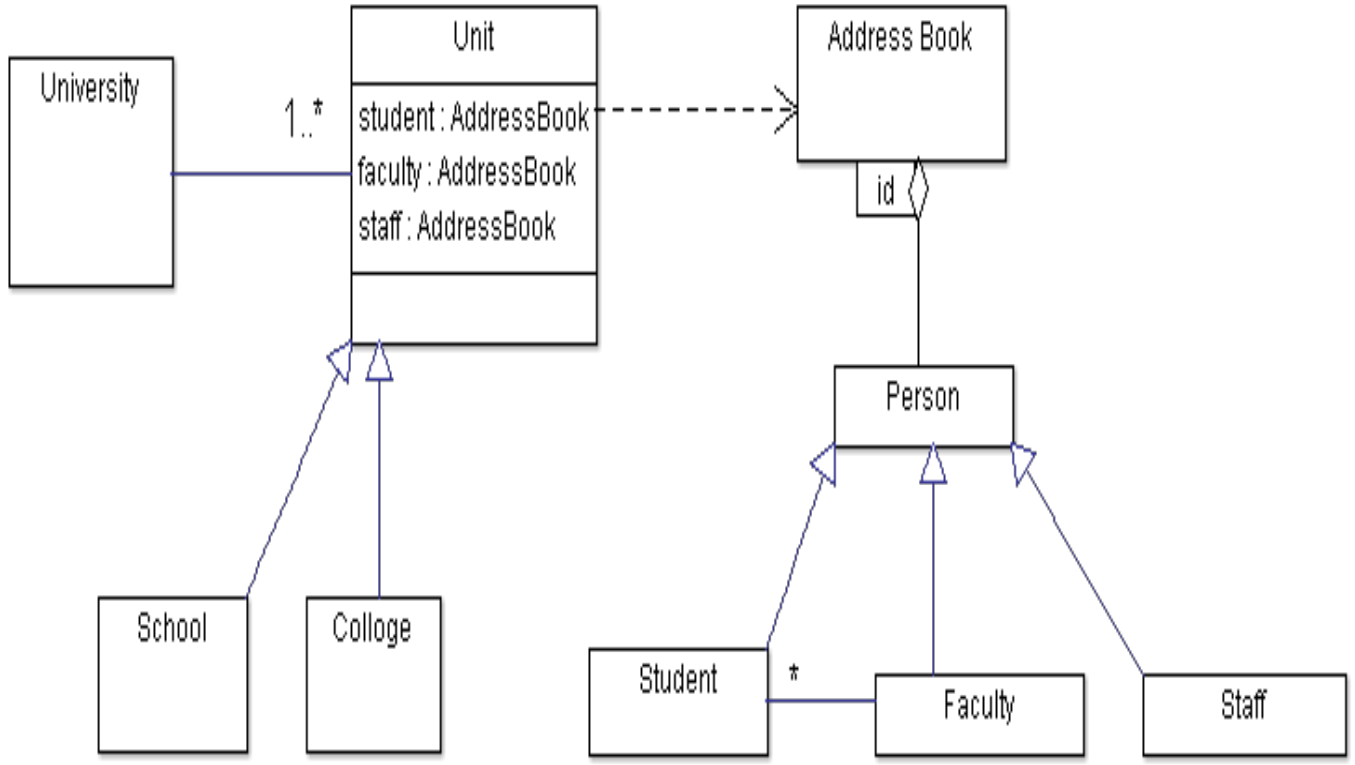
Bir sınıfın alt sınıflarının birden fazla ve çeşitli davranışlar göstermesidir.

UML Sembolleri



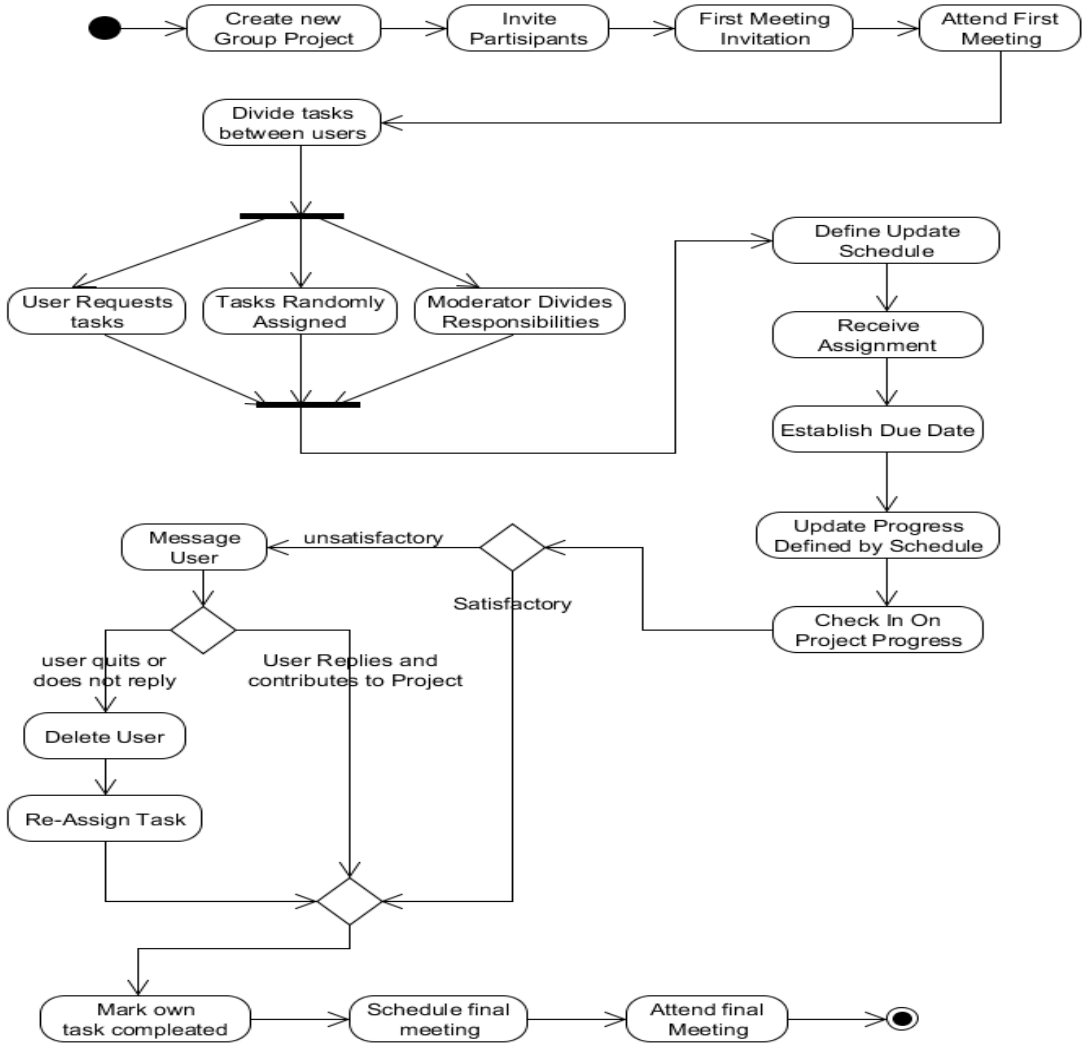
UML: Yapısal Diagramlar

- Sınıf (class)
- paket (package),
- Nesne (object),
- bileşen (component)
- bileşik yapı (composit structure)
- uygulama (deployment).



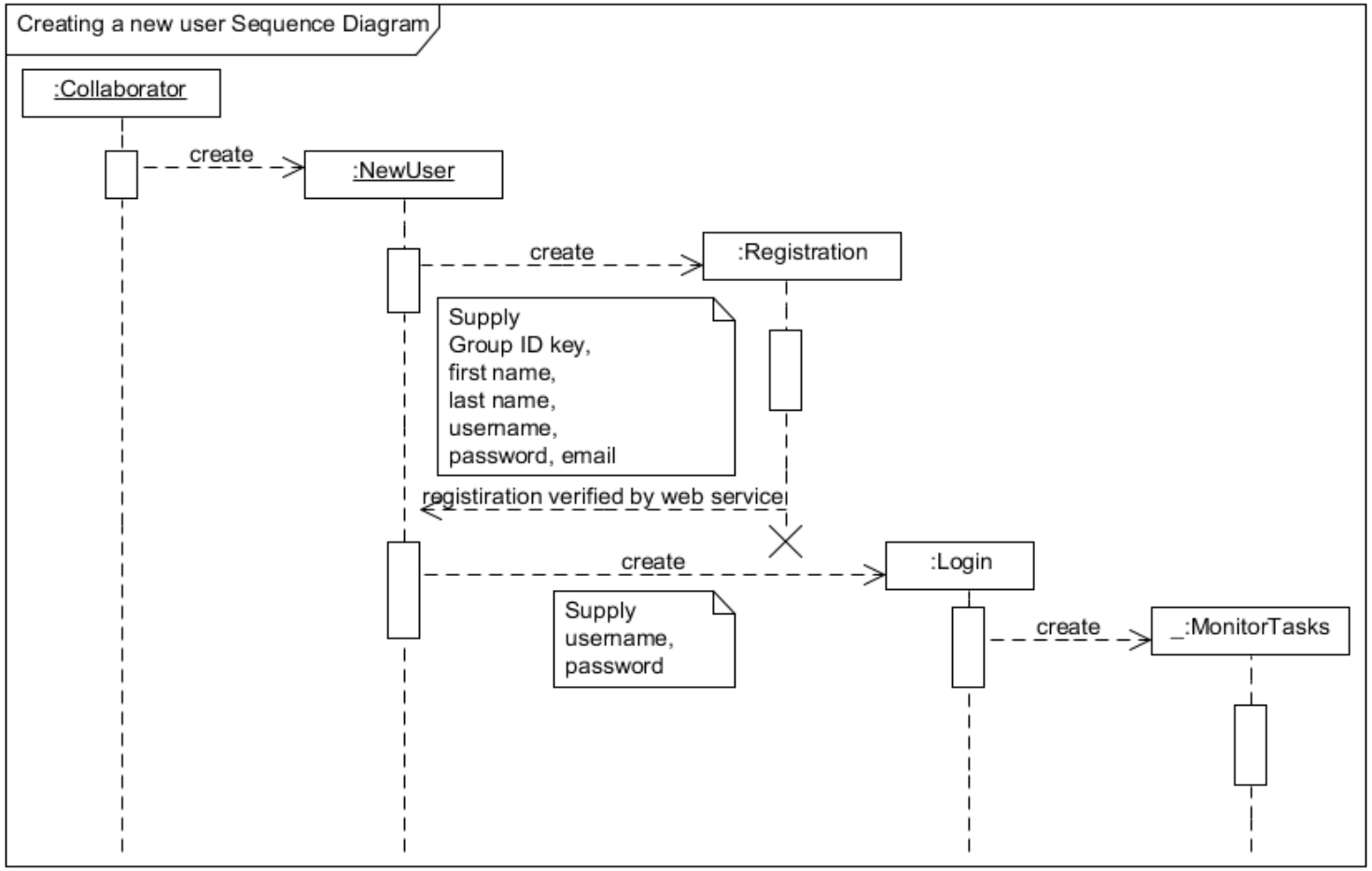
UML: Davranışsal Diagramlar

Aktivite (Activity) diagramı



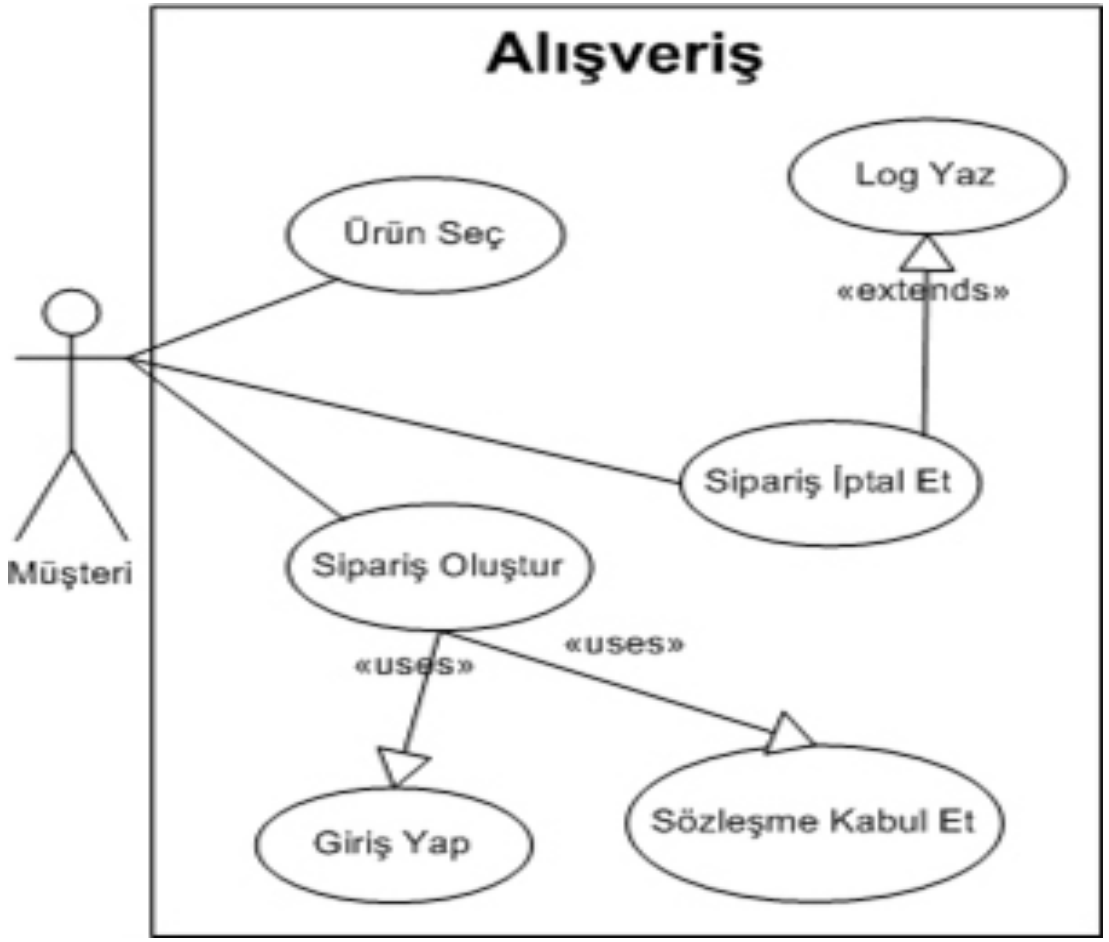
UML: Davranışsal Diagramlar

Sıralı (sequence) diagram



UML: Davranışsal Diagramlar

Kullanım senaryosu (Use case) diagram



Bölüm Soruları

1. Varlık , ilişki ve ilişki kümesi kavramlarını açıklayınız ve her birine örnek veriniz.
2. İs-a ilişkisi nedir? örnek şekil çizerek açıklayınız.
3. Many-to-many ilişkisini örnek vererek açıklayınız.
4. Özelleştirme, genelleme ve kümeleme kavramlarını acıklayınız.
5. Sınıf hiyerarşisi nedir?