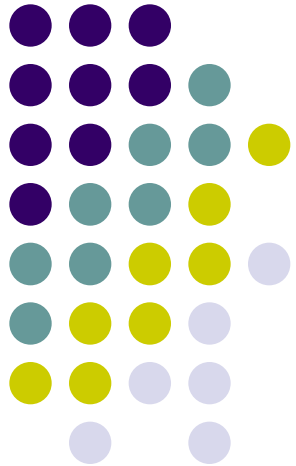
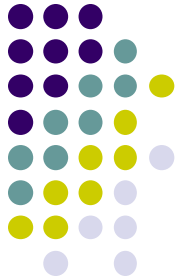
EBE-368 Veri Tabanı Yönetim Sistemleri

İlişkisel Model
(The Relational Model)

Dr. Dilek Küçük

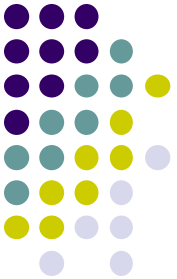


İlişkisel Model [1]



- Günümüzde en yaygın kullanılan veri tabanı modelidir.
  - Bir **ilişkisel veri tabanı** bir veya data fazla **ilişki**den (relation) oluşur.
  - İlişkiler iki kısımdan oluşur:
    - **İlişki şeması (relation schema)**
      - İlişkinin adını, alanlarının adlarını ve bu alanların veri türleri.
    - **İlişki örneği (relation instance)**
      - Şemaya karşılık gelen ve satırları (kayıtları) olan bir tablo

İlişkisel Model [2]



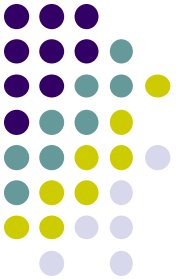
- İlişki şeması:

  Öğrenci (**no**:tamsayı, **ad**:karakter dizisi, **soyad**: karakter dizisi, **yaş**:tamsayı, **ortalama**:reel sayı)

- İlişki örneği:

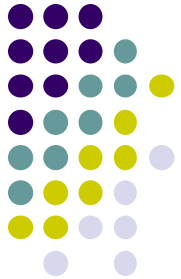
no	ad	soyad	yas	ortalama
21325	Necla	Yılmaz	18	2,9
21345	Öykü	Okan	20	2,1
21378	Elçin	Demir	19	3,3
21389	Özgür	Kara	19	3,1
21370	Serkan	Özdemir	20	1,8

İlişkisel Model [3]



- Bir örnekteki herhangi iki kayıt birbirinin aynı olamaz.
- **Alan kısıtlamaları** (domain constraints) her kolonun hangi veri kümesinden değer alabileceğini belirtir.
- Bir ilişkideki kolon sayısına **derece** adı verilir.
  - Öğrenci ilişkisinin derecesi 5'tir.

SQL Sorgu Dili



- 19 yaşındaki tüm öğrencileri bulmak için:

```
SELECT  *
FROM Ogrenci
WHERE yas = 19;
```

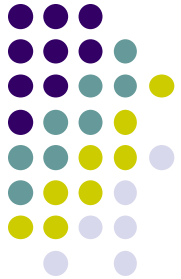
no	ad	soyad	yaş	ortalama
21378	Elçin	Demir	19	3,3
21389	Özgür	Kara	19	3,1

- 19 yaşındaki öğrencilerin sadece ad ve soyadları için:

```
SELECT  ad, soyad
FROM Ogrenci
WHERE yas = 19;
```

ad	soyad
Elçin	Demir
Özgür	Kara

SQL'de İlişki Oluşturma



- Ogrenci ilişkisini oluşturmak için:

```
CREATE TABLE Ogrenci
( no INTEGER,
  ad CHAR(30),
  soyad CHAR(30),
  yas INTEGER,
  ortalama REAL);
```

- VTYS kayıt eklenirken ve güncellenirken alanların veri türlerinin uygun olmasını zorlar.

SQL'de İlişki Silme ve Değiştirme

- Ogrenci ilişkisini silmek için aşağıdaki komut kullanılır. Şema bilgisi ve kayıtlar silinir.

  DROP TABLE Ogrenci;

- Ogrenci ilişkisine yeni bir alan eklemek için:

  ALTER TABLE Ogrenci

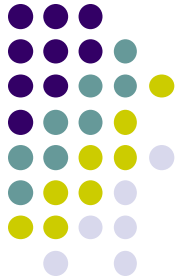
  ADD COLUMN sinif INTEGER;

- Bu alanı kaldırmak için:

  ALTER TABLE Ogrenci

  DROP COLUMN sinif;

SQL'de Kayıt Ekleme/Silme/Güncelleme



- Ogrenci ilişkisine kayıt eklemek için

  INSERT INTO Ogrenci (no, ad, soyad, yas, ortalama)
  VALUES (21325, 'Necla', 'Yılmaz', 18, 2.9);

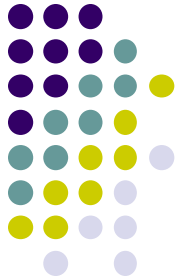
- Ogrenci ilişkisinden belirli bir şartı sağlayan kayıtları silmek için (ör. Adı 'Necla' olanları):

  DELETE FROM Ogrenci
  WHERE ad = 'Necla';

- Ogrenci ilişkisinde bir kaydı güncellemek için:

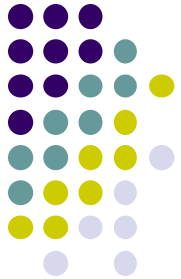
  UPDATE Ogrenci SET yas = 19
  WHERE no = 21325;

Bütünlük Kısıtlamaları (Integrity Constraints)



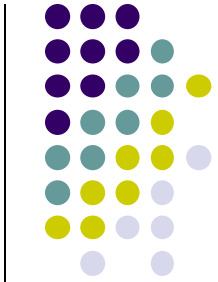
- Bir veri tabanı şemasında belirtilen ve veri tabanının bir örneğinde saklanabilecek veriyi sınırlayan durumlara **bütünlük kısıtlamaları** denir.
  - Veritabanı tanımlanırken ifade edilirler.
  - İlişkiler değiştirilirken kontrol edilirler.
- Veri tabanı yönetim sistemleri, veri tabanının bütünlük kısıtlamalarına uygun olmasını sağlar.

Anahtar Kısıtlamaları [1]



- Bir ilişkinin alanlarının belirli en küçük bir alt kümesinin bu ilişkideki kayıtları diğerlerinden ayırt etmesi ifadesidir.
  - Bir kaydı diğerlerinden ayıran bu tip bir alt kümeye **aday anahtar** veya **anahtar** adı verilir.
  - Bir ilişki örneğindeki iki farklı kayıt, anahtar alanlarında tamamen aynı değerlere sahip olamaz.
  - Anahtarın alanlarının hiçbir alt kümesi bir kayıt için ayırt edici değildir.
  - Bir anahtar içeren alan kümesine **süper anahtar** adı verilir.

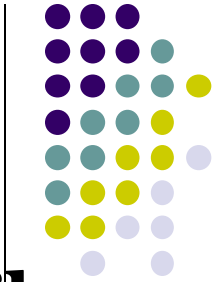
Anahtar Kısıtlamaları [2]



- Aday anahtarlar arasından biri **birincil anahtar (primary key)** olarak belirtilir.
- SQL'de anahtar oluşturan kümeler **UNIQUE** kelimesi ile belirtilebilir.
- Bu adaylardan en fazla biri **PRIMARY KEY** kelimesi ile birincil anahtar olarak ifade edilir.

```
CREATE TABLE Ogrenci (no INTEGER,
ad CHAR(30),
soyad CHAR(30),
yas INTEGER,
ortalama REAL,
UNIQUE (ad, soyad),
CONSTRAINT OgrenciAnahtari PRIMARY KEY (no));
```

Yabancı Anahtar Kısıtlamaları [1]



- İki ilişkiyi içeren en yaygın kısıtlama **yabancı anahtar (foreign key)** kısıtlamasıdır.
  - DersKaydi(ogrenci_no:tamsayı, ders:karakter dizisi, harf_notu:karakter)
  - **DersKaydi** ilişkisinin **ogrenci_no** alanı bir yabancı anahtardır ve **Ogrenci**'nin **no** alanına işaret eder.

ogrenci_no	ders	harf_notu
21325	VTYS111	A
21389	TARIH300	B
21378	TURKCE222	A

no	ad	soyad	yas	ortalama
21325	**Necla**	**Yılmaz**	**18**	**2,9**
21345	Öykü	Okan	20	2,1
21378	**Elçin**	**Demir**	**19**	**3,3**
21389	**Özgür**	**Kara**	**19**	**3,1**
21370	Serkan	Özdemir	20	1,8

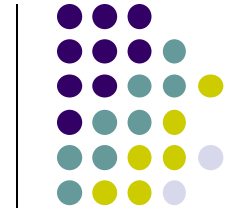
Yabancı Anahtar Kısıtlamaları [2]

- **CREATE TABLE DersKaydi**
  (ogrenci_no INTEGER,
  ders CHAR(20),
  harf_notu CHAR(1),
  PRIMARY KEY (ogrenci_no, ders),
  FOREIGN KEY (ogrenci_no) REFERENCES Ogrenci
  ON DELETE CASCADE
  ON UPDATE NO ACTION);
- Alan, birincil anahtar ve yabancı anahtar kısıtlamaları dışında da genel kısıtlamalar olabilir:
  - Tablo kısıtlamaları
  - Koşullar (assertions)

Anahtar Kısıtlamaları – Notlar [1]

- Bilinmeyen veya mevcut olmayan tablo alanlarına **null** değeri eklenebilir.
- **UNIQUE** anahtar kelimesini gibi null değer almaması gereken alanlar da **NOT NULL** ile kısıtlanabilir.
- Birincil anahtar kısıtlaması; UNIQUE ve NOT NULL kısıtlamalarının beraber zorunlu tutar.
  - Birincil anahtara ait alanlar null değer içeremez.

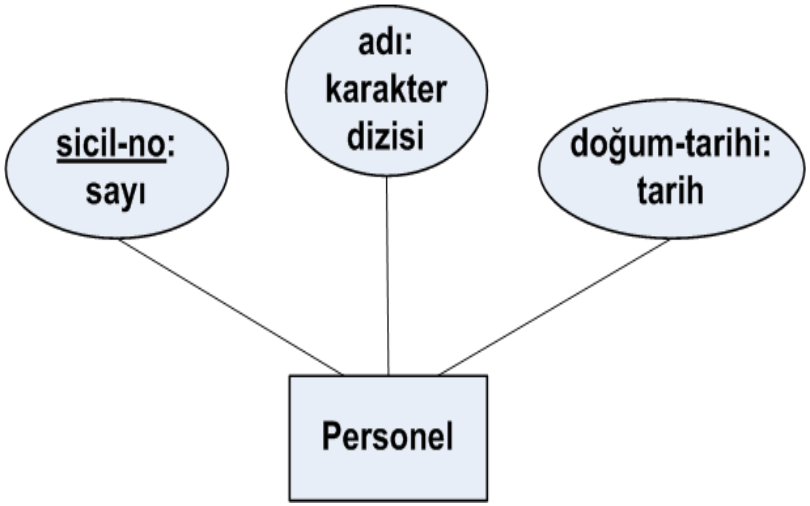
Anahtar Kısıtlamaları – Notlar [2]



- Yabancı anahtarlara ait alanlar null değer içerebilir.
- Yabancı anahtarın işaret ettiği alanın/alanların işaret edilen tabloda anahtar olması/olmaları gerekir.
- Bir yabancı anahtar kendi tablosundaki bir alana/alanlara da işaret edebilir.
- Yabancı anahtarın kısıtlaması durumunda işaret edilen tabloda veri silinmesi/güncellenmesi durumunda aşağıdaki seçeneklerden biri belirtilir:
  - **RESTRICT**
  - **NO ACTION**
  - **CASCADE**
  - **SET NULL**
  - **SET DEFAULT**

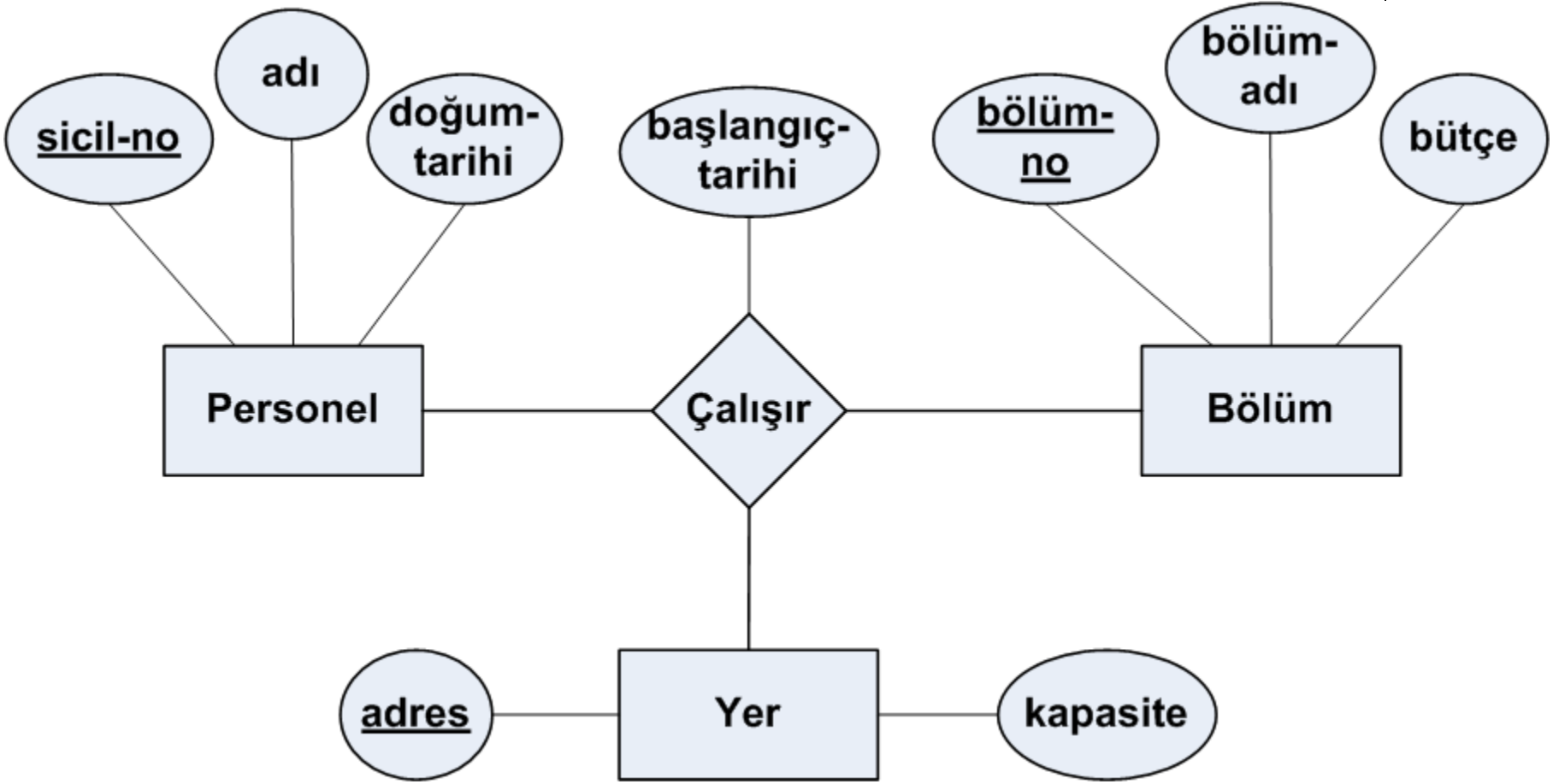
Mantıksal Veritabanı Tasarımı: Varlık-İlişki'den İlişkisele

- Varlık kümeleri tablolara dönüştürülür, her öznitelik tablonun bir alanı olur.

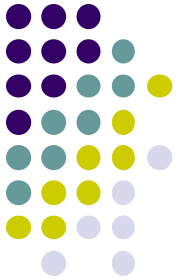



```
CREATE TABLE Personel
(sicil_no INTEGER,
  adi CHAR(30),
  dogum_tarihi DATE,
  PRIMARY KEY (sicil_no));
```

Mantıksal Veritabanı Tasarımı [2]



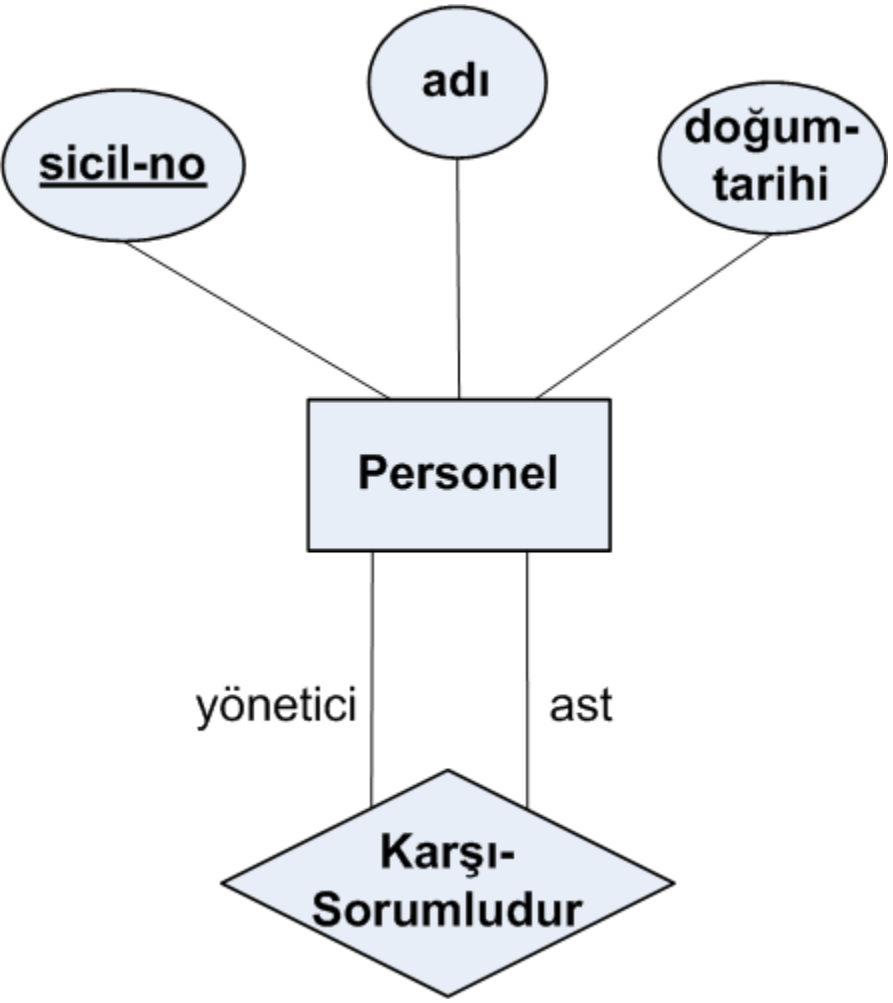
Mantıksal Veritabanı Tasarımı [3]



- İlişki kümeleri de tablolara dönüştürülür. Alanlar aşağıdakileri içerir:
  - Katılan varlıkların birincil anahtar öznitelikleri yabancı anahtar olarak,
  - İlişki kümesinin tanımlayıcı öznitelikleri.

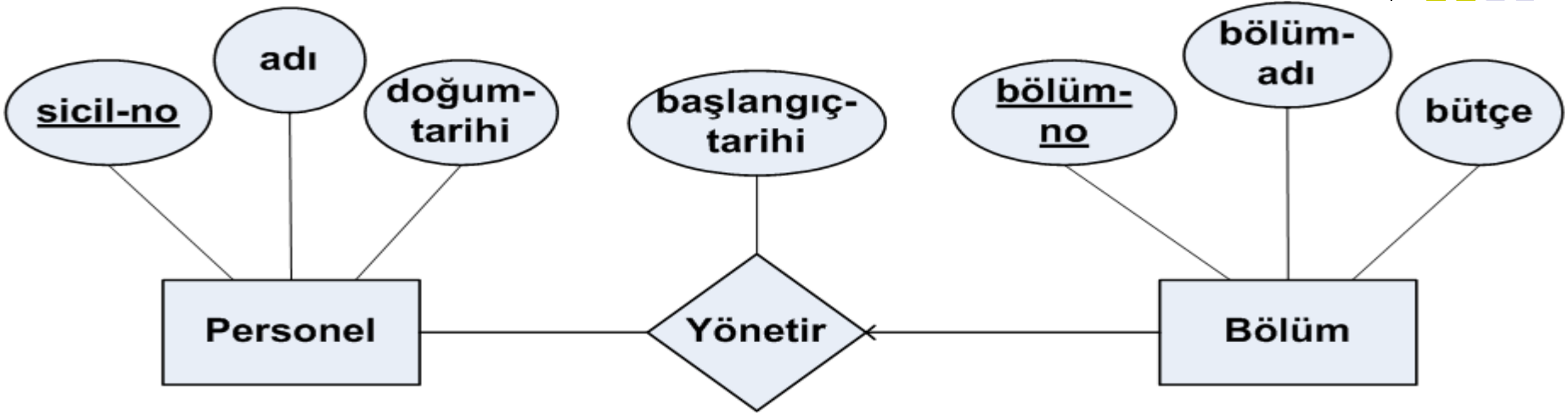
```
CREATE TABLE Calisir
(sicil_no INTEGER,
 bolum_no INTEGER,
 adres CHAR(20),
 baslangic_tarihi DATE,
 PRIMARY KEY (sicil_no,
   bolum_no, adres),
 FOREIGN KEY (sicil_no)
   REFERENCES Personel,
 FOREIGN KEY (bolum_no)
   REFERENCES Bolum,
FOREIGN KEY (adres)
   REFERENCES Yer);
```

Mantıksal Veritabanı Tasarımı [4]




```
CREATE TABLE
    Karsi_Sorumludur
(yonetici_sicil_no INTEGER,
 ast_sicil_no INTEGER,
PRIMARY KEY
          (yonetici_sicil_no,
          ast_sicil_no),
 FOREIGN KEY
          (yonetici_sicil_no)
   REFERENCES Personel
                    (sicil_no),
FOREIGN KEY (ast_sicil_no)
   REFERENCES Personel
                    (sicil_no));
```

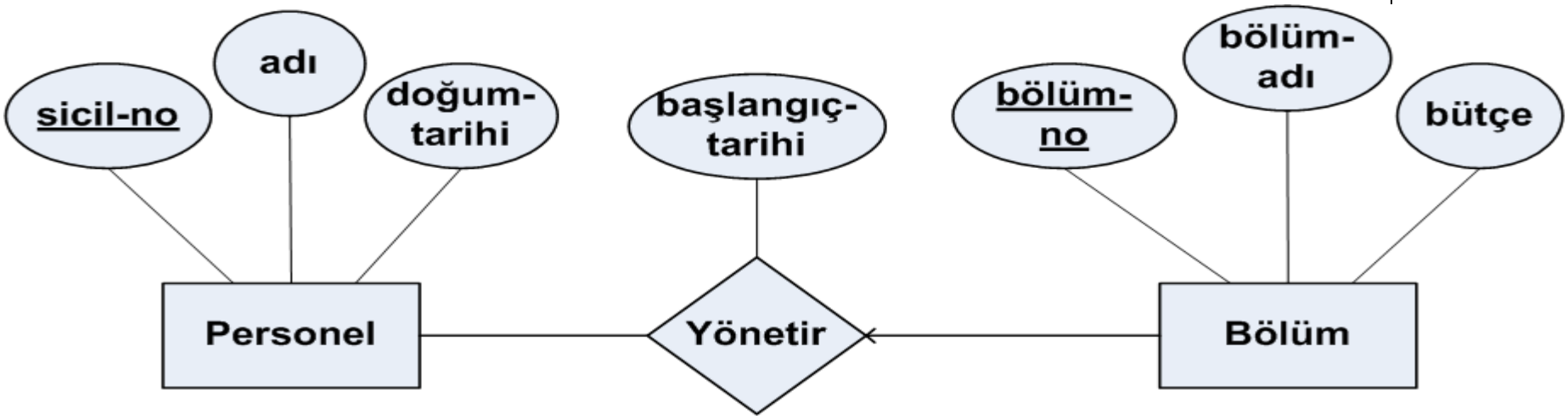
Mantıksal Veritabanı Tasarımı [5]



1. Yaklaşım (toplam 3 tablo)

```
CREATE TABLE Yonetir
(sicil_no INTEGER,
 bolum_no INTEGER,
 baslangic_tarihi DATE,
 PRIMARY KEY (bolum_no),
 FOREIGN KEY (sicil_no) REFERENCES Personel,
FOREIGN KEY (bolum_no) REFERENCES Bolum);
```

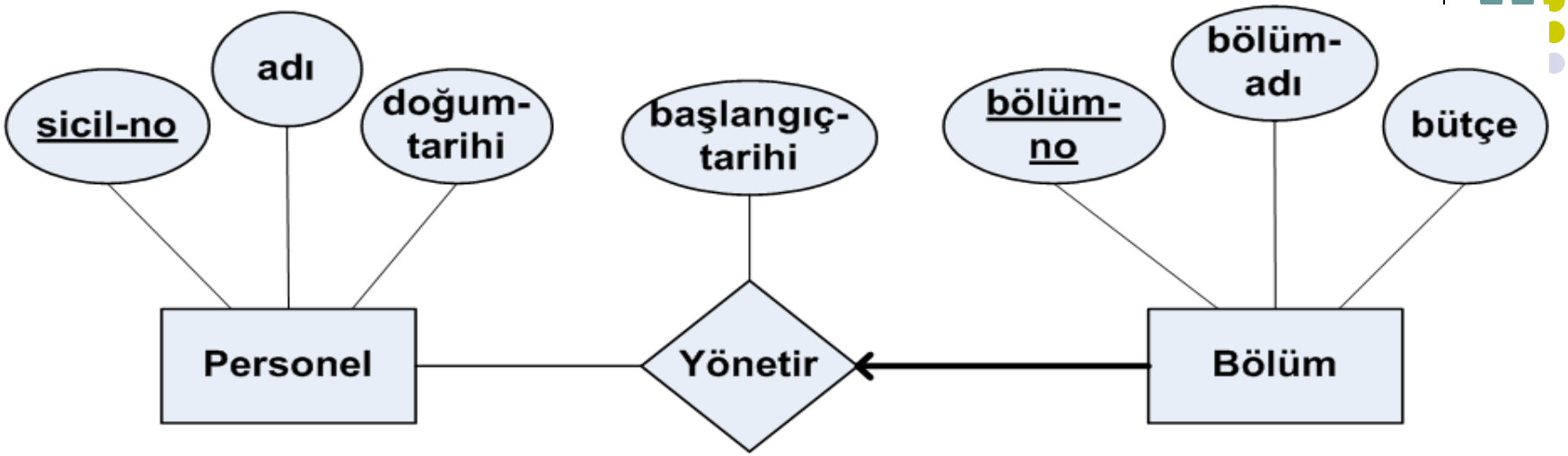
Mantıksal Veritabanı Tasarımı [6]




```
CREATE TABLE Bolum_Yonetim
(bolum_no INTEGER,
  bolum_adi CHAR(20),
  butce REAL,
  sicil_no INTEGER,
  baslangic_tarihi DATE,
  PRIMARY KEY (bolum_no),
  FOREIGN KEY (sicil_no) REFERENCES Personel);
```

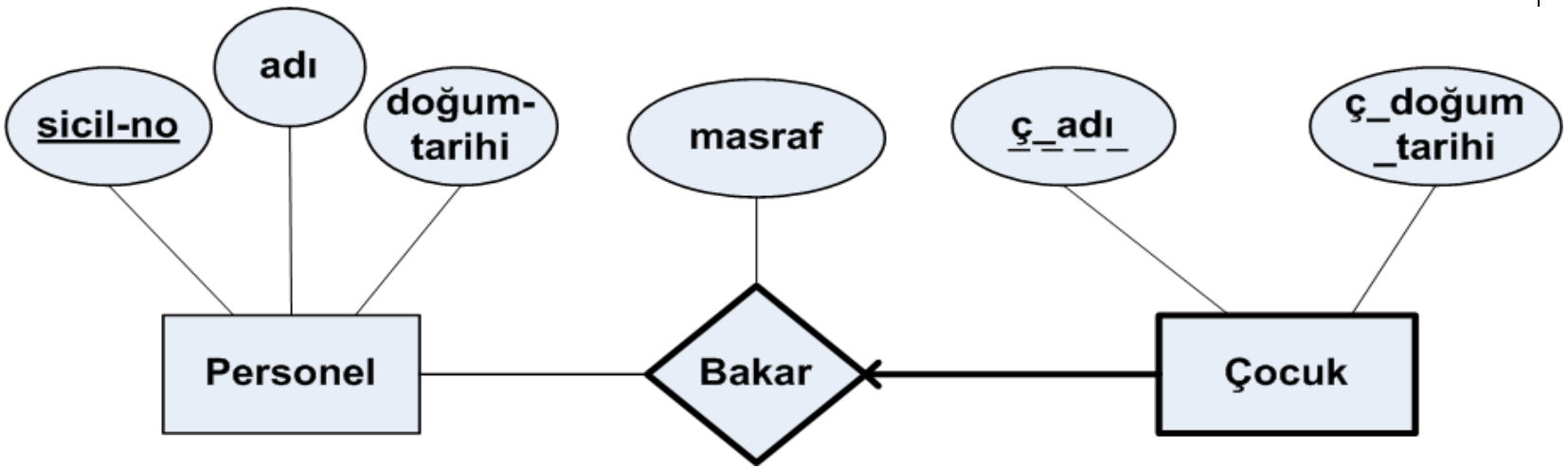
2. Yaklaşım (toplam 2 tablo)

Mantıksal Veritabanı Tasarımı [7]



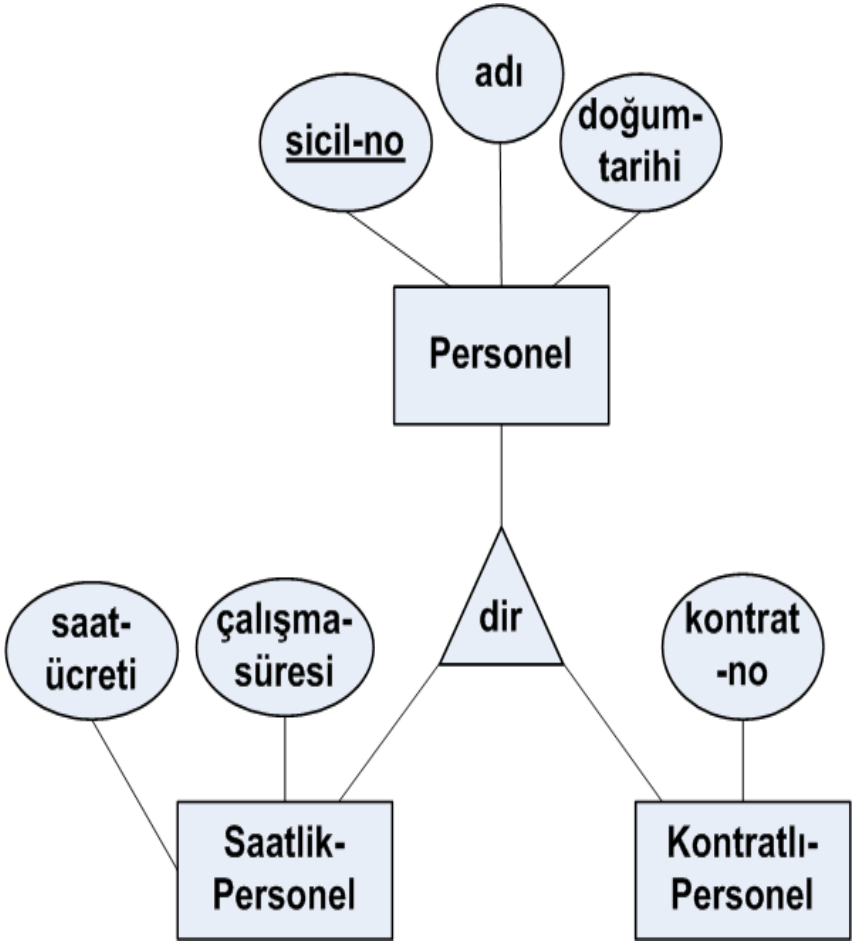

```
CREATE TABLE Bolum_Yonetim
(bolum_no INTEGER,
 bolum_adi CHAR(20),
 butce REAL,
 sicil_no INTEGER NOT NULL,
 baslangic_tarihi DATE,
 PRIMARY KEY (bolum_no),
 FOREIGN KEY (sicil_no) REFERENCES Personel
                        ON DELETE NO ACTION);
```

Mantıksal Veritabanı Tasarımı [8]




```
CREATE TABLE Cocuk_Bakim
(c_adi CHAR(20),
  c_dogum_tarihi DATE,
  masraf REAL,
  sicil_no INTEGER,
  PRIMARY KEY (c_adi, sicil_no),
  FOREIGN KEY (sicil_no) REFERENCES Personel
                         ON DELETE CASCADE);
```

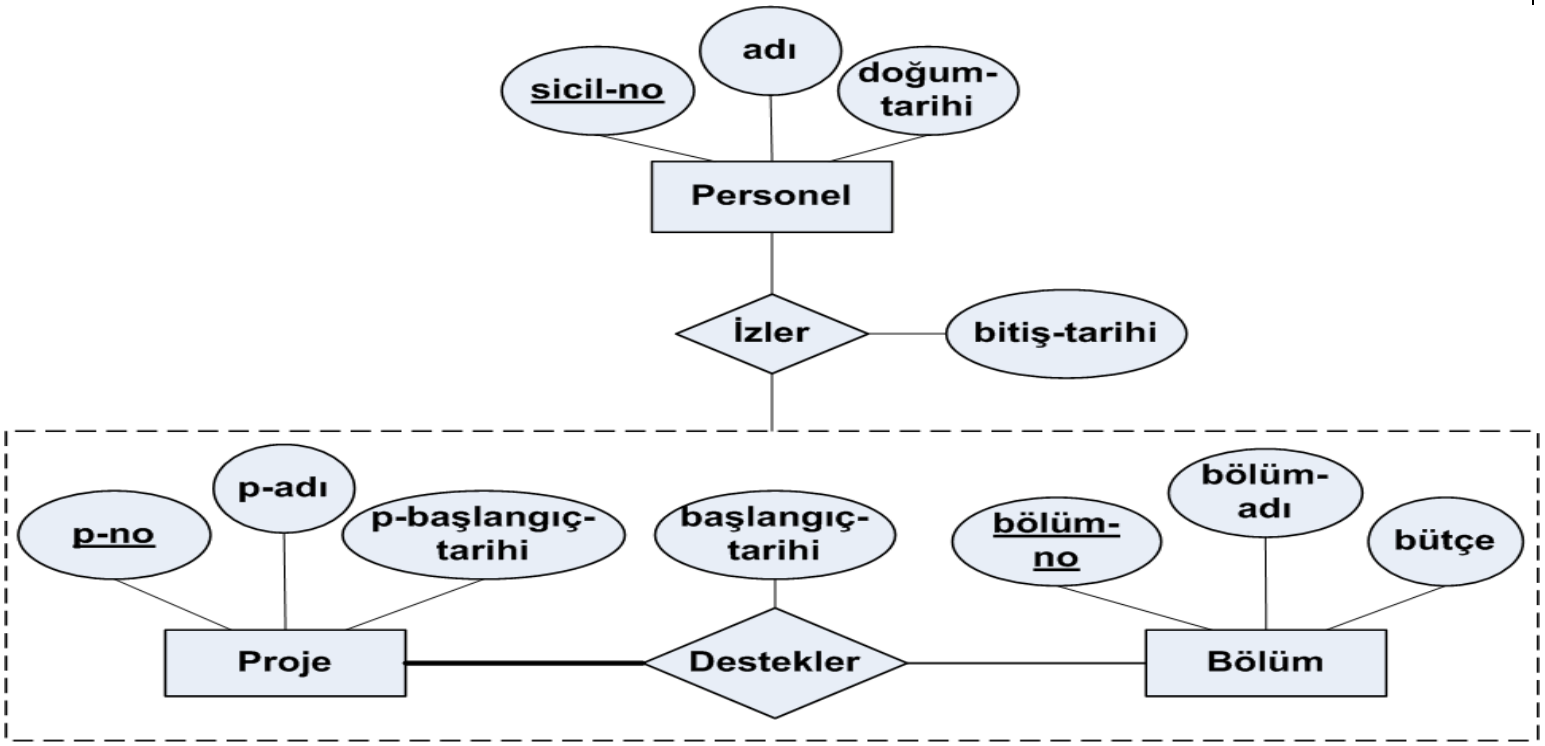
Mantıksal Veritabanı Tasarımı [9]



İki yaklaşım mevcuttur:

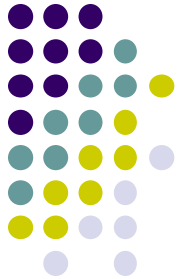
- Her varlık için ayrı tablo oluşturmak.
  - Alt sınıflara ait tablolar sicil_no yabancı anahtarıyla Personel tablosuna işaret eder.
- Sadece alt sınıflara ait iki tablo oluşturmak.
  - Bu tablolar varlıkların hem kendi özniteliklerine hem de Personel'in özniteliklerine karşılık gelen alanlar içerirler.

Mantıksal Veritabanı Tasarımı [10]



- İzler için sicil_no, p_no, bolum_no, ve bitis_tarihi alanlarından oluşan bir tablo oluşturulabilir.
- Eğer Destekler ilişkisinin hiç özniteliği olmasaydı ve İzler ilişkisine tam katılımı olsaydı Destekler için ayrı bir tablo oluşturulmasına gerek olmazdı.

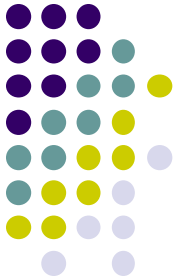
Görüntüler



- Satırları gerçekte veri tabanında depolanmayan, gerektikçe tanımı kullanılarak hesaplanan tablolara **görüntü (view)** adı verilir.

```
CREATE VIEW B-Ogrenci (ad, soyad, ogrenci_no, ders)
        AS SELECT O.ad, O.soyad, O.ogrenci_no, D.ders
                FROM Ogrenci O, DersKaydi D
                WHERE O.ogrenci_no = D.ogrenci_no AND
                        D.harf_notu = 'B';
```

- Görüntüler mantıksal veri bağımsızlığını ve veri güvenliğini sağlamada faydalıdırlar.



Teşekkürler