

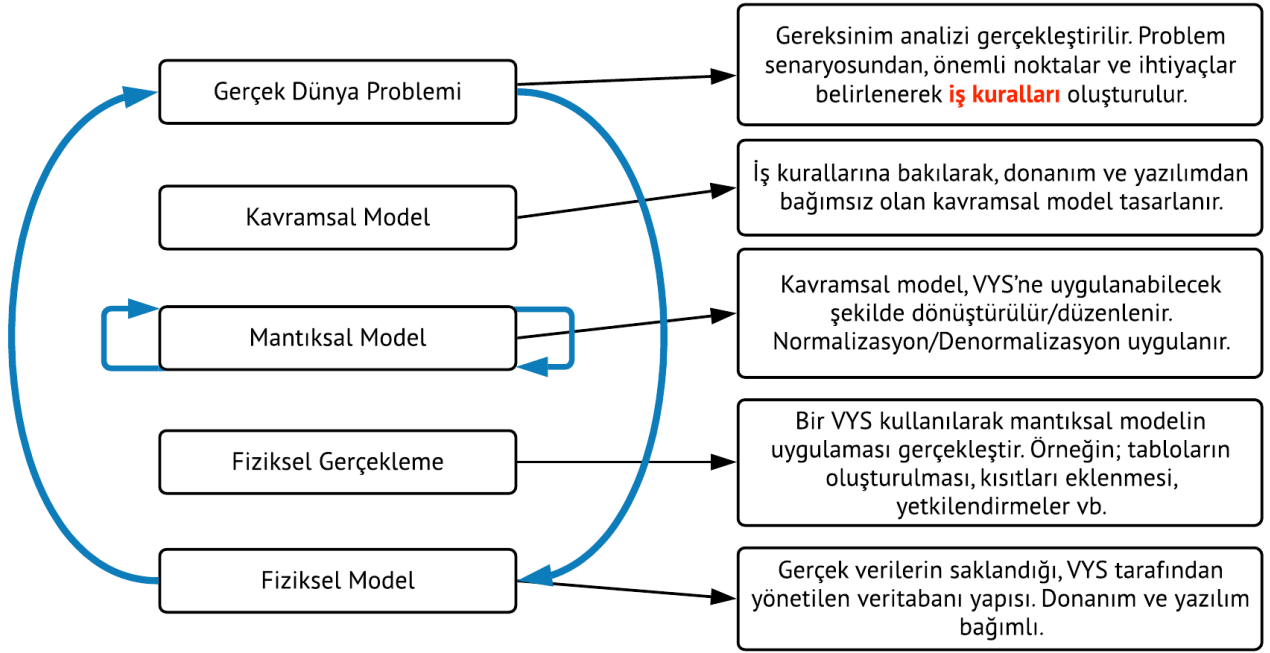
BSM211 Veritabanı Yönetim Sistemleri - Celal ÇEKEN, İsmail ÖZTEL, Veysel Harun ŞAHİN

İş Kuralları, Veri Modelleri, Varlık Bağintı Modeli

Konular

- Veritabanı Geliştirme Yaşam Döngüsü
- İş Kuralları (Business Rules)
- Veri Modeli Temel Bileşenleri
- İş Kurallarını Veri Modeline Dönüştürme
- Veri Modellerinin Gelişimi
 - Dosya Sistemi
 - Hiyerarşik Model
 - Ağ Modeli
 - İlişkisel Model
 - Varlık Bağintı Modeli
 - Nesne Yönelimli Model
 - Yeni Veri Modelleri
- Veri Soyutlama
- Varlık Bağintı Modeli
 - Temel Kavramlar
 - Varlıklar Arası Bağıntılar
 - Var Olma Bağımlılığı (Existence Dependency)
 - Zayıf/Güçlü Bağıntılar
 - Bağintı Dereceleri
 - Varlık Bağintı Modeli (VBM/ERM) ile Tasarım
- Kaynaklar

Veritabanı Geliştirme Yaşam Döngüsü



İş Kuralları (Business Rules)

- Veritabanı (varlık, nitelik, bağıntı ve kısıtlar) oluşturulurken iş kurallarına bakılır.
- İş kuralı:** Veritabanı tasarımı yapılacak organizasyon ile ilgili işleyiş, kural ya da yönetmeliğin özetlenmiş şekline iş kuralları denilebilir. İş kuralları ihtiyaç listesine benzer.
- Örnek iş kuralları:
 - Bir müşteri çok sayıda sipariş verebilir.
 - Her müşterinin adı, soyadı, telefon numarası vs. istenir.
 - Öğrenciler bir ara sınav ve bir yarıyıl sonu sınavına girerler.
- İş kurallarının kaynağı, son kullanıcılar, yöneticiler, kural koyucular ve yazılı dokümanlar (standart, yönetmelik vs.) olabilir.
- İş kurallarını oluşturmak için doğrudan son kullanıcılarla görüşmek oldukça etkili bir çözümdür.
- Veritabanı tasarımı açısından iş kurallarının önemi;
 - Kullanıcılar ile tasarımcılar arasındaki iletişimi sağlar.
 - Tasarımcının verinin doğasını, önemini ve kapsamını anlamasını sağlar.
 - Tasarımcının iş süreçlerini anlamasını sağlar.
 - Tasarımcının doğru bir veri modeli geliştirmesine yardım eder (veriler arası ilişkiler ve kısıtların kolayca belirlenmesini sağlar).
 - Kuruluşun veriye bakışını standart haline getirir.
- İş kuralları oluşturulduktan sonra, gerçekleştirilecek veritabanının modellenmesi aşamasına geçilir.

Veri Modeli

- Veri modeli:** Karmaşık gerçek dünya veri yapılarının basit olarak gösterilmesi (genellikle grafiksel) için kullanılan araca veri modeli ismi verilir.
- Veri modeli, veritabanı tasarımcıları, uygulama programcıları ve son kullanıcılar arasındaki iletişimi kolaylaştırır.
- Veri modelleri sayesinde veritabanı tasarımını gerçekleştirmek daha kolay olur.

- Veri modelleme yinelemeli (iterative) bir işlemdir. Önce basit model oluşturulur. Daha sonra ayrıntılar eklenir. En sonunda veritabanı tasarımında kullanılan şablon (blueprint) elde edilir.

Veri Modelinin Temel Bileşenleri

- **Varlık (Entity):** Hakkında veri toplanan ve saklanan her şey (öğrenci, ders, personel vb.). Gerçek dünyadaki nesneleri ifade eder. Var olan ve benzerlerinden ayırt edilebilen her şey.
- **Varlık kümesi (Entity set):** Aynı türden varlıkların oluşturduğu kümeye denir (Öğrenciler, Dersler vb.).
- **Nitelik (Attribute):** Varlığın sahip olduğu özellikler.
- **Bağıntı (Relationship):** Varlıklar arasındaki ilişkiyi ifade eder.
 - Bir-Çok (One to Many 1:M)
 - Bir müşteri çok sayıda sipariş verebilir.
 - Her sipariş yalnızca bir müşteri tarafından verilir.
 - Çok-Çok (Many to Many M:N)
 - Bir öğrenci çok sayıda ders alabilir.
 - Her ders çok sayıda öğrenci tarafından alınabilir.
 - Bir-Bir (One to One 1:1)
 - Bir mağaza bir personel tarafından yönetilir.
 - Bir personel bir mağazayı yönetir.
- **Kısıtlar (Constraints):** Veri üzerindeki sınırlamalardır. Veri bütünlüğünün sağlanması açısından önemlidir. Örneğin;
 - Öğrenci notunun 0-100 arasında olması
 - T.C. kimlik numarasının 11 karakter olması
 - Aynı ürünün birden fazla kayıt edilememesi

Veri Modellerinin Gelişimi

- Dosya Sistemi
- Hiyerarşik Model
- Ağ Modeli
- İlişkisel model
- Varlık Bağlantı modeli
- Nesne Yönelimli Model
- Yeni Veri Modelleri

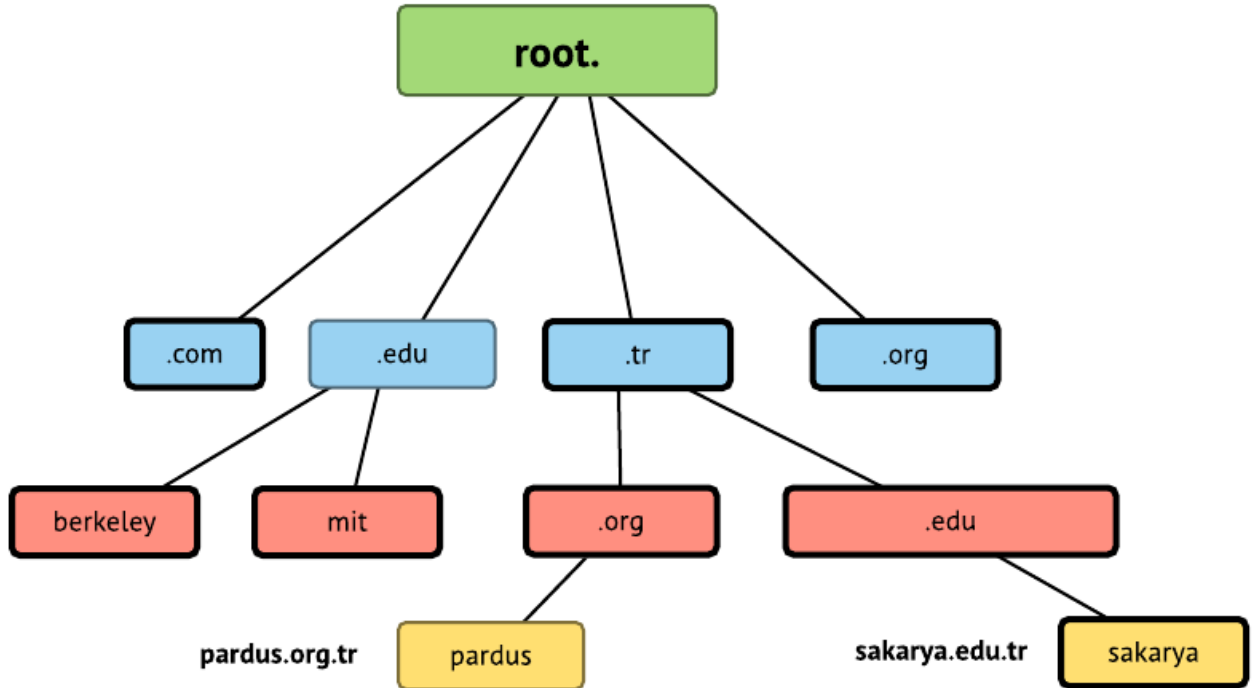
Dosya Sistemi

- 1960-1970 lerde çoğunlukla IBM ana çatı (mainframe) sistemlerde kullanılmıştır
- Dosyalar arasında ilişki yoktur.
- Veri tekrarı ve sıralı tarama gibi sorunları bulunmaktadır.
- Örnek bir dosya yapısı aşağıda görünmektedir.

Ders Kodu	Ders Adı	Öğr.Türü	Dönem	Bölüm Adı
BSM207	VERİ YAPILARI (B)	1. Öğretim	1	BİLGİSAYAR MÜHENDİSLİĞİ PR.
BSM303	VERİTABANI YÖNETİM SİSTEMLERİ (A)	1. Öğretim	1	BİLGİSAYAR MÜHENDİSLİĞİ PR.
BSM207	VERİ YAPILARI (B)	2. Öğretim	1	BİLGİSAYAR MÜHENDİSLİĞİ PR. (İÖ)
BSM303	VERİTABANI YÖNETİM SİSTEMLERİ (A)	2. Öğretim	1	BİLGİSAYAR MÜHENDİSLİĞİ PR. (İÖ)
BSM303	VERİTABANI YÖNETİM SİSTEMLERİ (?)	Uzaktan Eğitim	1	BİLGİSAYAR MÜHENDİSLİĞİ PR. (UZAKTAN EĞİTİM)
EBT514	VERİTABANI TASARIM VE YÖNETİMİ (?)	Uzaktan Eğitim	1	BİLİŞİM TEKNOLOJİLERİ PR. (YL) (UZAKTAN EĞİTİM)
BSM829	UZMANLIK ALANI (?)	1. Öğretim	1	BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ PR. (YL)
BSM929	UZMANLIK ALANI (?)	1. Öğretim	1	BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ PR. (DR)
BSM401	BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI (F)	1. Öğretim	1	BİLGİSAYAR MÜHENDİSLİĞİ PR.
BSM401	BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI (F)	2. Öğretim	1	BİLGİSAYAR MÜHENDİSLİĞİ PR. (İÖ)

Hiyerarşik Model

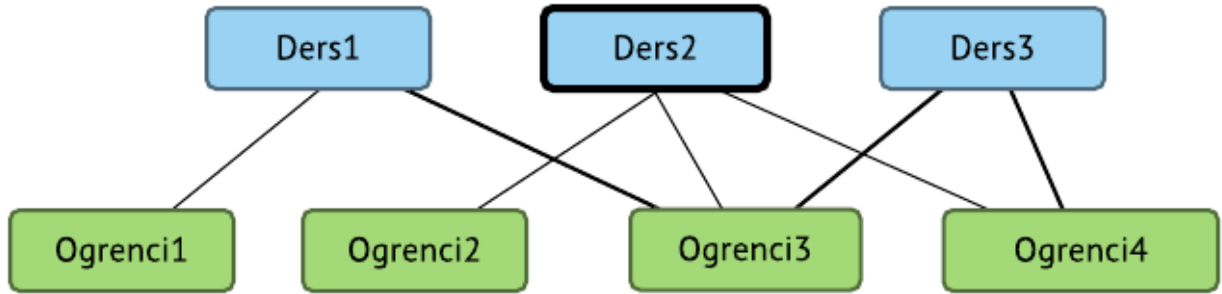
- 1960'larda büyük miktardaki verileri yönetebilmek için geliştirilmiştir.
- Veriler ağaç yapısı şeklinde organize edilir.
- Ana-çocuk (parent-child) arasında 1:M ilişkisi vardır. Kayıtların sadece 1 ana (parent) kaydı vardır.



Ağ Modeli

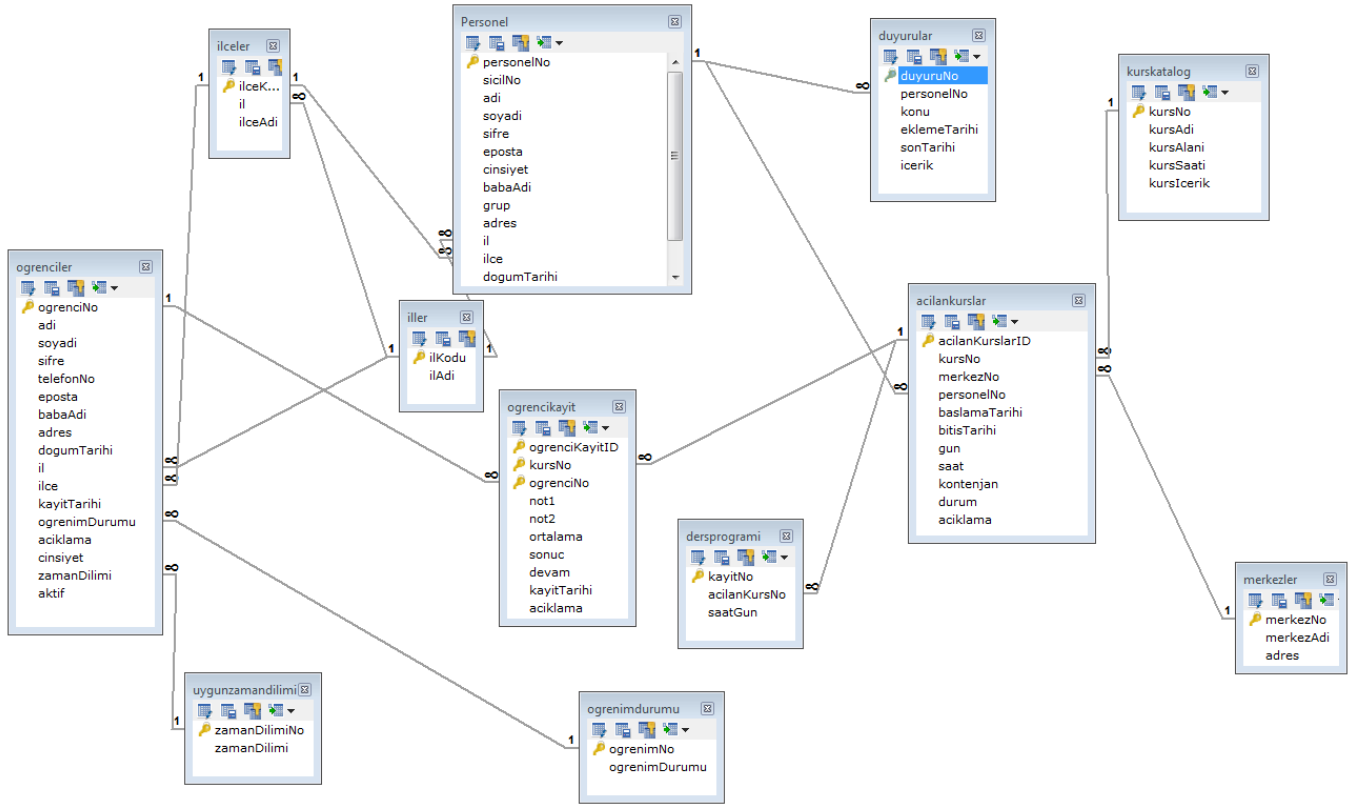
- 1970'lerde geliştirilmiştir. Veritabanı başarımını artırmak üzere daha karmaşık ilişkilere izin verilir.
- Hiyerarşik modelden farklı olarak kayıtların birden fazla ana (parent) kayıtları olabilir.
- Ağ modeliyle birlikte ortaya çıkan ve hala kullanılan bazı kavramlar aşağıdadır.
 - **Şema:** Tüm veritabanının, veritabanı yöneticisi tarafından görünen kavramsal organizasyonu.
 - **Alt şema:** Veritabanının istenen bilgiyi üreten uygulama programı tarafından görünen kısmı.

- **Veri işleme dili (data manipulation language, DML):** Veritabanında bulunan verilerin, sorgulama işlemleri yapılarak güncellenmesi, yeni verilerin eklenmesi ve olan verilerin silinme işlemlerinin yapılmasını sağlayan dil.
- **Veri tanımlama dili (data definition language, DDL):** Veritabanında bulunan verilerin tip, yapı ve kısıtlamalarının tanımlanmasını sağlayan dil.
- **Anlık sorgu (ad hoc query):** Yazılımlarla birlikte gelmeyen kullanıcının kendi oluşturduğu sorgulara verilen isimdir.
- Ağ modelinin dezavantajı, çok basit sorgular için bile karmaşık program kodlarının kullanımını gerektirmesidir.



İlişkisel Model

- 1970’de E. F. Codd tarafından ortaya atılmıştır (A Relational Model of Data for Large Shared Databanks, Communications of the ACM, June 1970, pp. 377–387).
- İlişkisel Veritabanı Yönetim Sistemleri (Relational Database Management Systems, RDBMS) tarafından kullanılır.
- RDBMS’nin en önemli özelliklerinden birisi ilişkisel modelin karmaşık yapısını kullanıcıdan gizlemesidir.
- Kullanıcı, ilişkisel modeli, verileri içeren tablolardan oluşan bir yapı gibi görür.
- Tablolar birbirlerine ortak alanlarla bağlanırlar.
- İlişkisel şema, varlıklar, varlıkların nitelikleri ve aralarındaki bağlantıların gösteriminden oluşur.

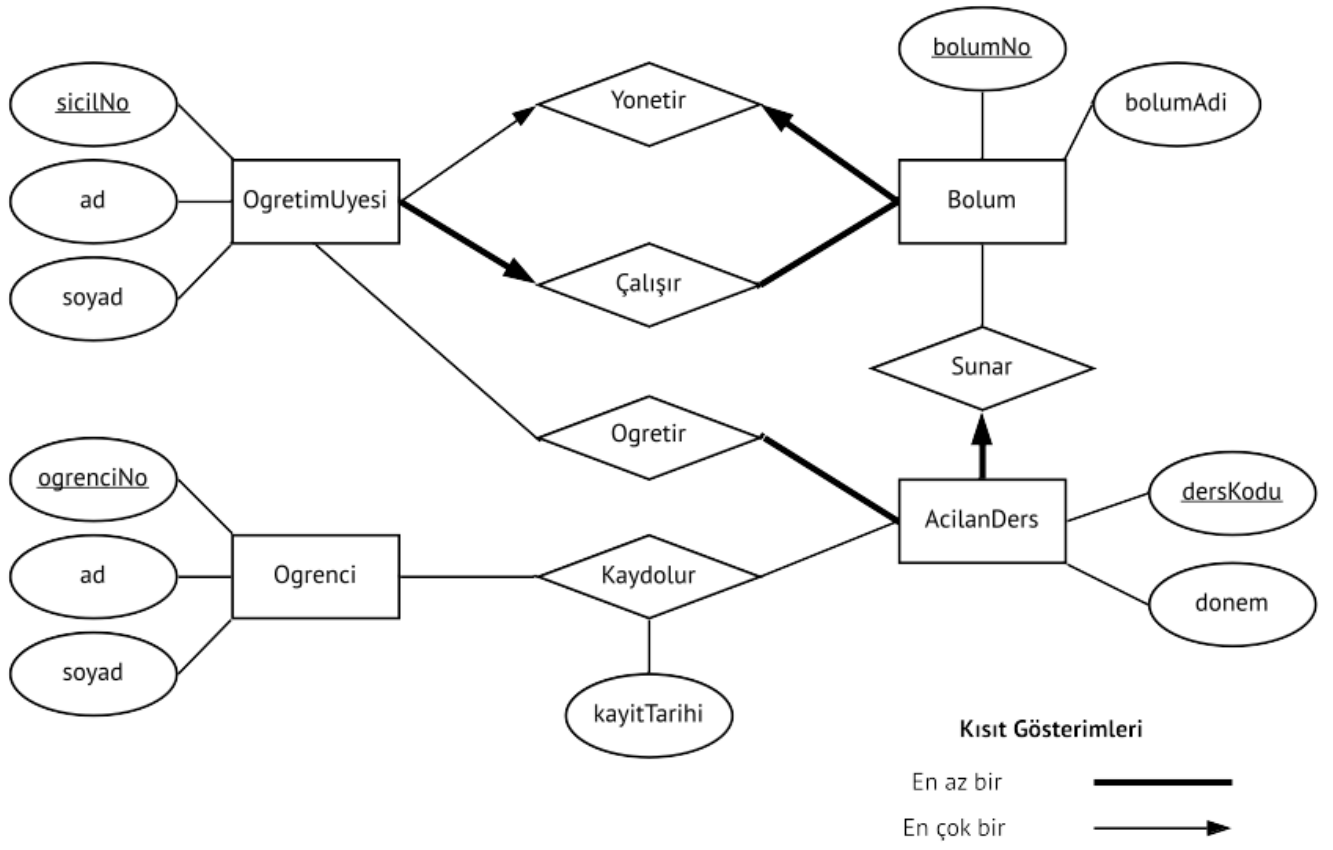


- İlişkisel veritabanı modelinin en güçlü yanlarından birisi, verileri yönetmek için yapısal sorgulama dili (structured query language, SQL) dilinin kullanılıyor olmasıdır.
- SQL dili nasıl yapılması gerektiğini anlatmak yerine ne yapılması gerektiğinin ifade edildiği basit bir dildir.
- Bu nedenle, SQL kullanılarak veri tabanlarının tasarımı ve yönetimi daha kolaydır.
- İlişkisel bir veritabanı yönetim sistemi 3 temel bileşenden oluşur.
 - Verilerin saklandığı veritabanı
 - SQL komutlarını derleyerek istenenleri gerçekleştiren SQL Motoru (SQL Engine)
 - Kullanıcılarla iletişimi sağlayan arayüzler.

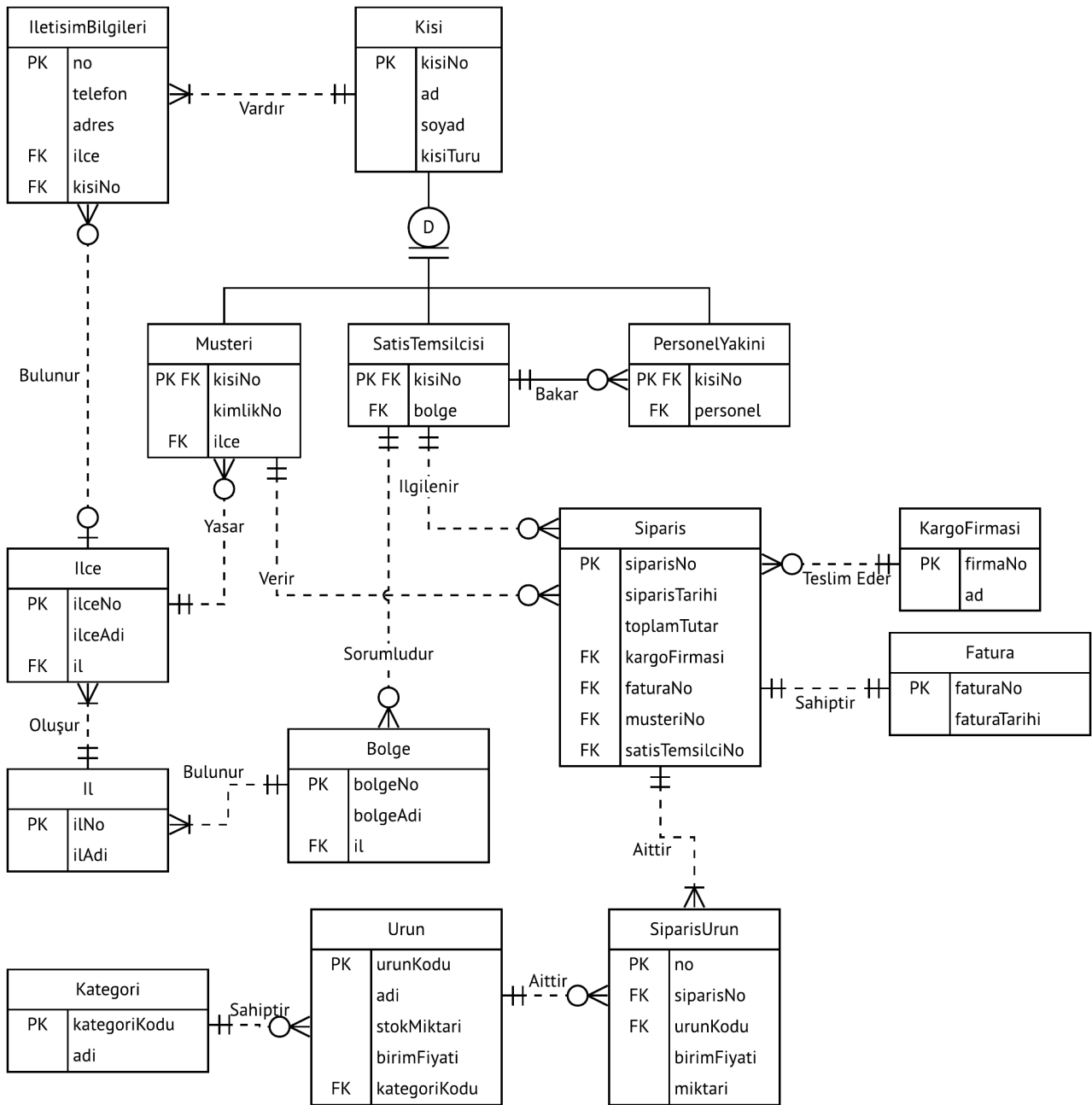
Varlık Bağlantı Modeli

- İlişkisel model daha önceki modellere göre çok daha kullanışlı olmasına rağmen veritabanı tasarımı için ilişkisel modelin grafiksel gösterimi olan varlık bağlantı modeli (VBM) (entity relationship model, ERM) daha sık kullanılır. 1976'da Peter Chen tarafından önerilmiştir.
- İlişkisel modelinin tamamlayıcısı olduğu için kullanımı oldukça yaygınlaşmıştır.
- i) Chen gösterimi ve ii) Crow's Foot gösterimi sıkça kullanılan gösterim şekillerindendir.
- Ders kapsamında Crow's Foot gösterimi kullanılacaktır.

Chen Gösterimi



Crow's Foot Gösterimi



Nesne Yönelimli Model

- Nesne yönelimli programlama paradigmasından esinlenerek geliştirilen modeldir.
- Varlık bağıntı (VB - ER) modelindeki varlık (entity) bu modelde nesne olarak adlandırılır.
- Nesne hakkındaki bilgi, VB modelindeki niteliklere karşılık gelir.
- Varlık kümesi sınıf olarak adlandırılır.
- VB modelinden farklı olarak sınıflar üye fonksiyonlara da sahiptirler. Kisi ara, Ad listele vb.

VB Modeli	İlişkisel Model	Nesne Yönelimli Model
Varlık	Kayıt/Satır	Nesne
Varlık Kümesi	Tablo	Sınıf
Nitelik	Kolon/Özellik	Değişken
-	S.Yordam/Fonksiyon	Yöntem

Yeni Veri Modelleri

- Nesne İlişkisel Model (Object/Relational Model)
 - İlişkisel modelle nesne yönelimli modelin birleştirilmesi sonucu ortaya çıkmıştır.
- Genişletilebilir İşaretleme Dili (Extensible Markup Language, XML)
 - Çoğunlukla, farklı platformlar arası veri değişimi için kullanılan veri tanımlama standardıdır. Yapısal olmayan verileri tanımlamak için de kullanılır.
- JavaScript Nesne Gösterimi (JavaScript Object Notation, JSON)
 - Çoğunlukla, farklı platformlar arası veri değişimi için kullanılan veri tanımlama standardıdır. Yapısal olmayan verileri tanımlamak için de kullanılır.
- NoSQL
 - İlişkisel modelin yetersiz kaldığı büyük hacimli verilerin yönetimi için tercih edilir.

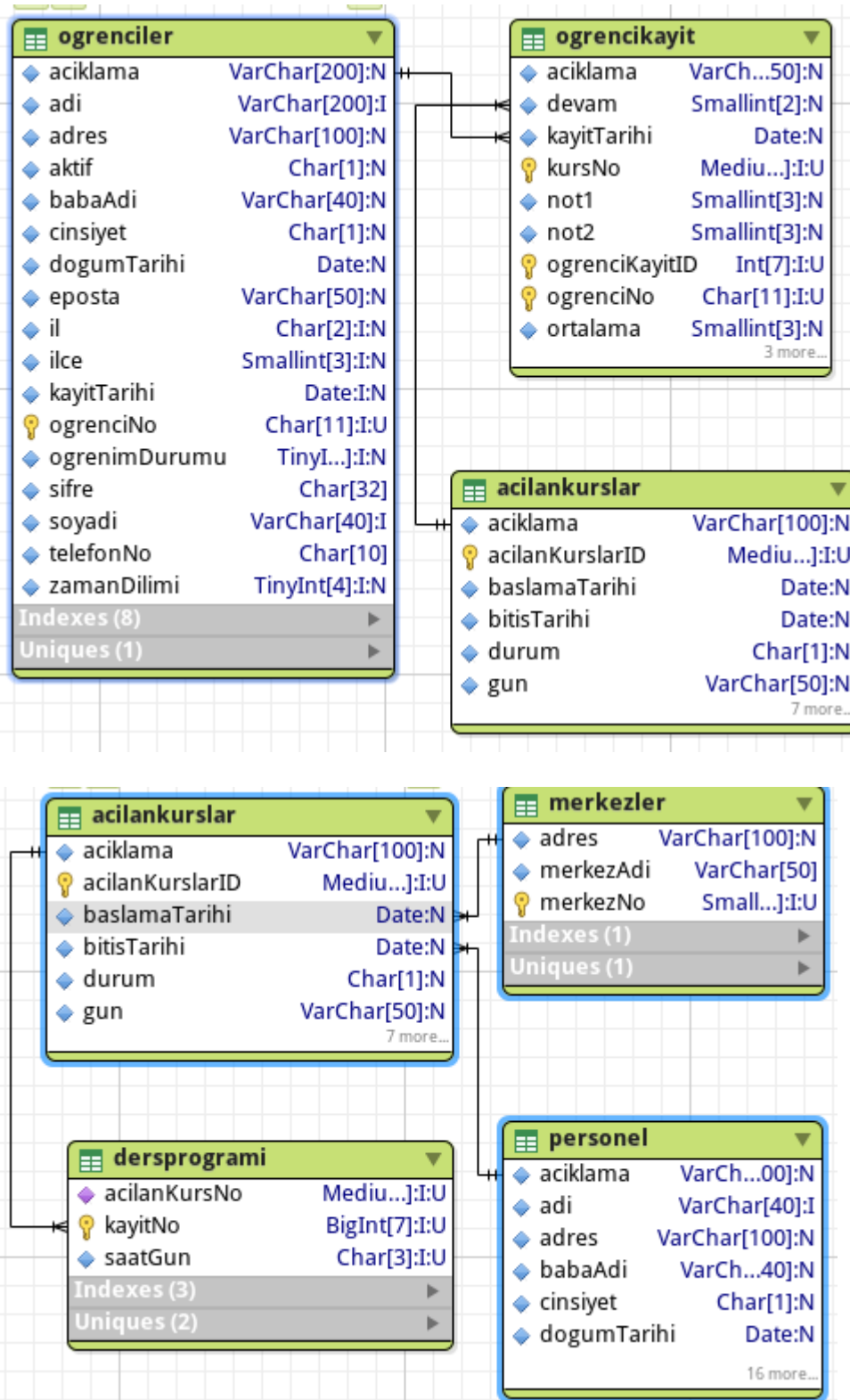
Veri Soyutlama

Veri modellerinin daha iyi anlaşılabilmesini sağlamak amacıyla ANSI-SPARC, 1970'lerin başında, veri soyutlamanın 3 düzeyini tanımlamıştır. (ANSI-SPARC: American National Standards Institute, Standards Planning and Requirements Committee.)

- Harici Model (External Model)
- Kavramsal Model (Conceptual Model)
- Dahili Model (Internal Model)

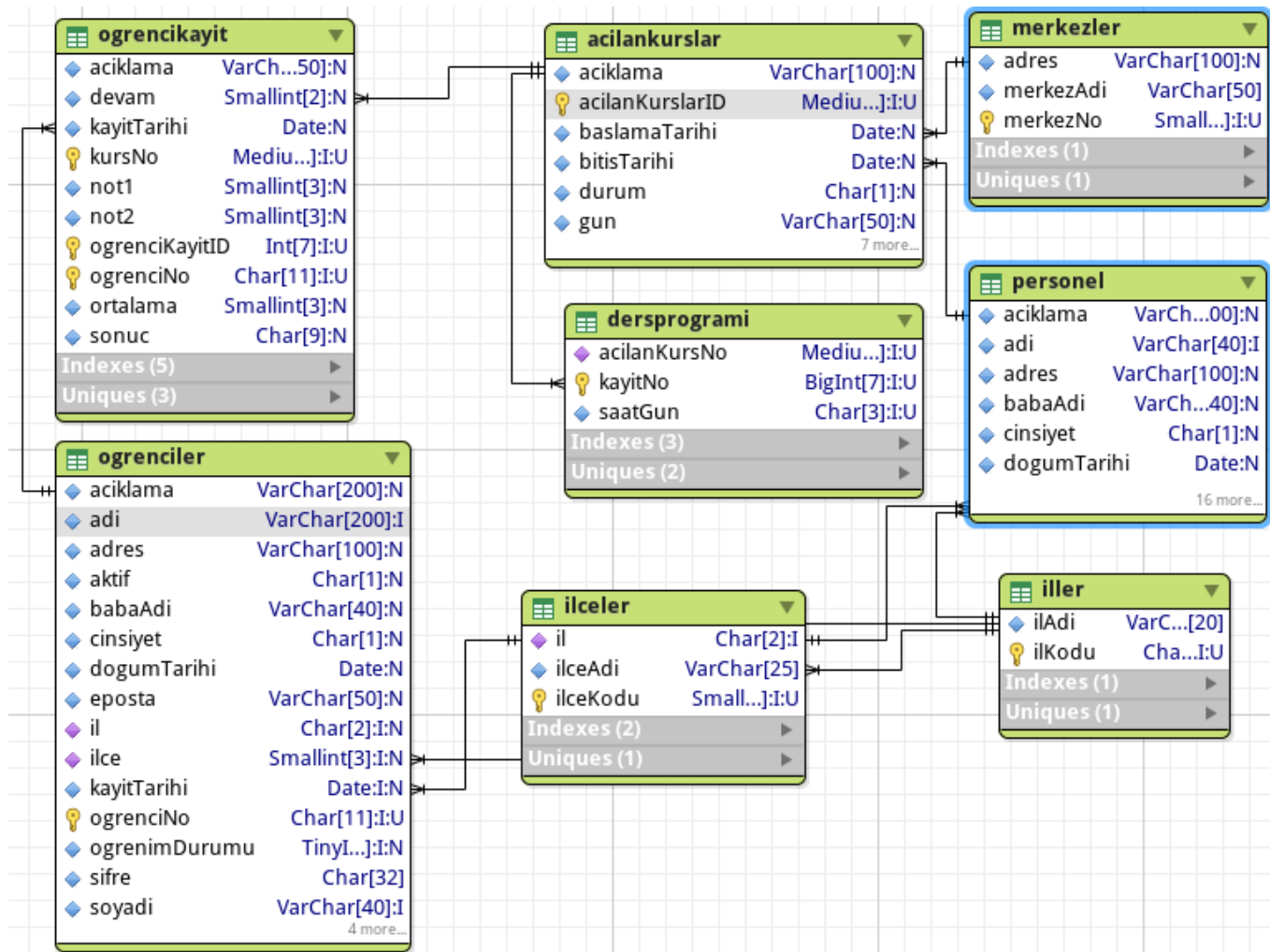
Harici Model (External Model)

- Veritabanının son kullanıcılar açısından görünen kısmı. Veritabanının sadece kullanıcıyla ilgili alt bölümlerini ifade eder.



Kavramsal Model (Conceptual Model)

- Veritabanının veritabanı tasarımcısı açısından görünen kısmı. Veritabanının tüm alt bölümlerini birleştirerek global olarak görünmesini sağlar.
- Varlık Bağlantı Diyagramı (VBD – ERD) ile gösterilir. Kullanılan yazılım (DBMS) ve donanımdan bağımsızdır. Donanım ya da yazılım değişikliği kavramsal model tasarımını etkilemez.
- Kavramsal model mantıksal görünüş olarak da kullanılır.



Dahili Model (Internal Model)

- Veritabanının, Veritabanı Yönetim Sistemi tarafından görünen kısmı.
- Dahili model = ilişkisel model
- Donanım bağımsız, yazılım bağımlı.

```
-- CREATE TABLE "dersprogrami" -----
CREATE TABLE `dersprogrami` (
  `kayitNo` BigInt( 7 ) UNSIGNED ZEROFILL AUTO_INCREMENT NOT NULL,
  `acilanKursNo` MediumInt( 6 ) UNSIGNED ZEROFILL NOT NULL UNIQUE,
  `saatGun` Char( 3 ) NOT NULL UNIQUE,
  PRIMARY KEY ( `kayitNo` ),
  CONSTRAINT `acilanKursNo` UNIQUE( `acilanKursNo`, `saatGun` ) )
ENGINE = InnoDB
AUTO_INCREMENT = 97;
-----

-- CREATE TABLE "duyurular" -----
CREATE TABLE `duyurular` (
  `duyuruNo` Int( 7 ) UNSIGNED AUTO_INCREMENT NOT NULL,
  `personelNo` Char( 11 ) NOT NULL,
  `konu` VarChar( 50 ) NOT NULL,
  `eklemeTarihi` Date NOT NULL,
  `sonTarihi` Date NOT NULL,
  `icerik` VarChar( 10000 ) NULL,
  PRIMARY KEY ( `duyuruNo` ) )
ENGINE = InnoDB
AUTO_INCREMENT = 6;
-----
```

Varlık Bağntı Modeli

Bu bölümde varlık bağntı modeli detaylı bir şekilde anlatılmaktadır.

Giriş

- Varlık Bağntı Diyagramı (VBD, Entity Relationship Diagram (ERD)) veritabanının kavramsal olarak modellenmesini sağlayan yazılım ve donanımlardan tamamen bağımsız gösterim şekilleridir.
- Farklı kullanıcılar arasında haberleşmeyi sağlamak için kullanılır.
- VB modeline bakarak veri tabanının tasarımını gerçekleştirmek daha kolaydır.
- VBD, veritabanının temel bileşenleri olan varlık, nitelik (özellik), varlıklar arası bağıntılar (relationship) ve kısıtlardan meydana gelen çizelgelerdir.
- Chen notasyonu kavramsal modellemeyi ön plana çıkartır.
- Crow's Foot notasyonu ise daha çok uygulama (implementasyon) yönelimli yaklaşımı ön plana çıkartır.
- UML notasyonu, hem kavramsal hem de uygulama (implementasyon) modelleme yöntemleri için kullanılabilir.

İş Kurallarını Varlık Bağntı Modeline Dönüştürme

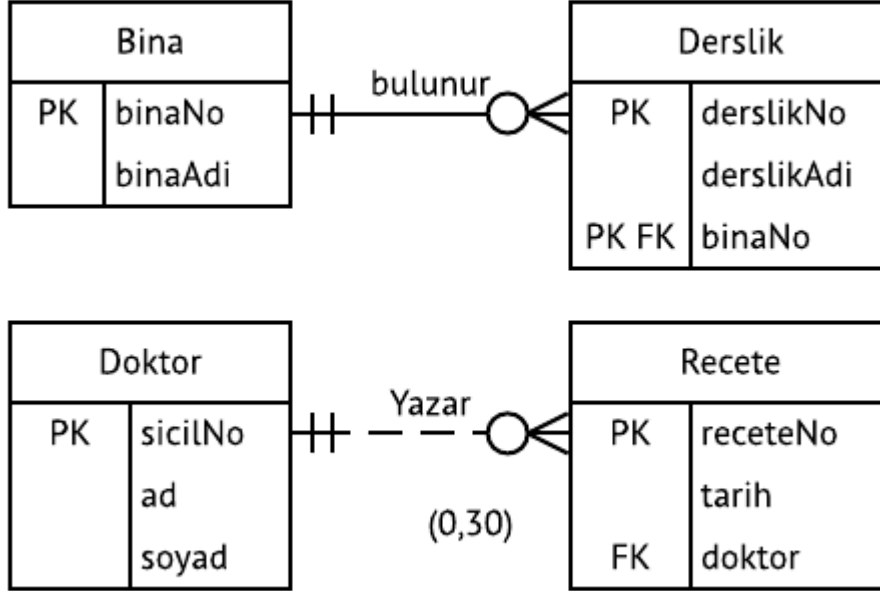
- Genel olarak iş kurallarındaki isimler varlık, fiiller ise varlıklar arasındaki bağıntı olma adaydır.
- Hakkında bilgi bulunan isim ya da isim tamlamaları varlık adayı iken, bilgi bulunmayanlar varlığa ait nitelik adaydır.
 - *Müşterinin* **ad, soyad, numara, adres** bilgileri saklanır.
 - Bir *müşteri* çok sayıda *fatıra* **üretir**.

- Bağlıntılar iki yönlüdür.
 - **1 öğretim üyesi çok** sayıda (4) **ders** verebilir
 - **1 ders** sadece **1 öğretim üyesi** tarafından verilebilir.
 - **1 kişi 1 bölüme** yönetici olabilir.
 - **1 bölüm** sadece **1 kişi** tarafından yönetilebilir.
 - **1 öğrenci çok** sayıda **derse** kayıt yaptırabilir
 - **1 ders çok** sayıda **öğrenci** tarafından alınabilir.
 - İsimlendirme kuralları
 - Kodlama türleri
 - Linux Coding Style, Linus Torvalds
 - Hungarian Notation,
 - GNU Coding Standards
 - Java Coding Style Guide
 - Birincil anahtar (primary key): Bir varlık kümesindeki her bir varlığı eşsiz bir şekilde tanımlamıza olanak tanıyan nitelik(ler).
 - Tek bir nitelik (alan) olabileceği gibi birden fazla niteliğin birleşiminden de oluşabilir.
 - urunKodu
 - dersKodu + ogrenciNo
 - Birleşik nitelikler:
 - adres: cadde, şehir, ülke, posta kodu vb.
 - Detaylı sorgular için basit özellikli alanlara dönüştürülmeliler.
 - Basit nitelikler
 - yaş, ad, soyad, cinsiyet vb.
 - Türetilmiş nitelikler
 - Değeri diğer özellikler kullanılarak oluşturulan özellikler.
- ```
SELECT AVG(AGE(dogumTarihi)) FROM Kisiler
```
- Saklanmalı mı, hesaplanmalı mı?
  - Saklanması Durumunda:
    - Avantaj: Az işlemci gücü gerekir, veriye daha hızlı erişim, geçmiş bilgisi için kullanılabilir
    - Dezavantaj: Güncel değer için sürekli denetlenmelidir, fazladan yer kaplar
  - Hesaplanması Durumunda:
    - Avantaj: Yer tasarrufu sağlar. Her an güncel değer olur.
    - Dezavantaj: Çok işlemci gücü gerekir, veriye daha yavaş erişim olur, sorgular daha karmaşık olur.

## Varlıklar Arası Bağlantılar

### Bir - Çok Bağıntısı

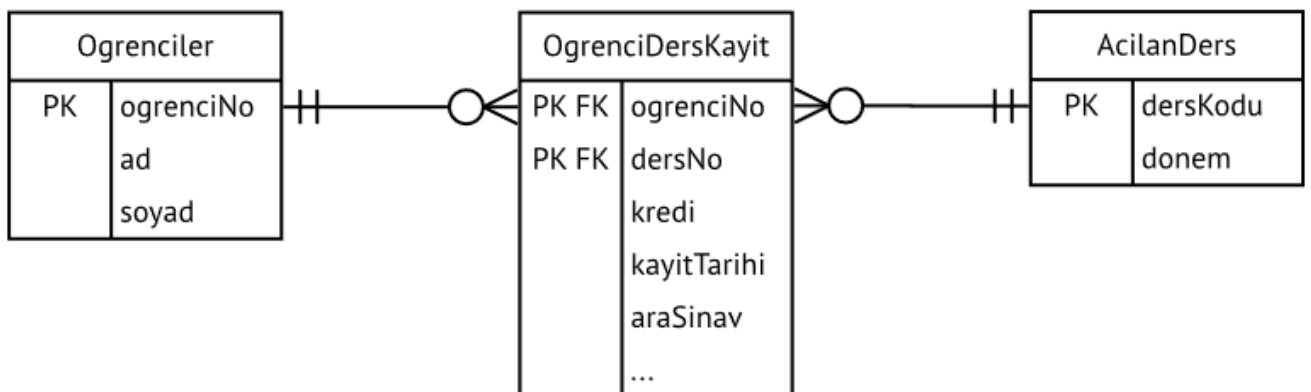
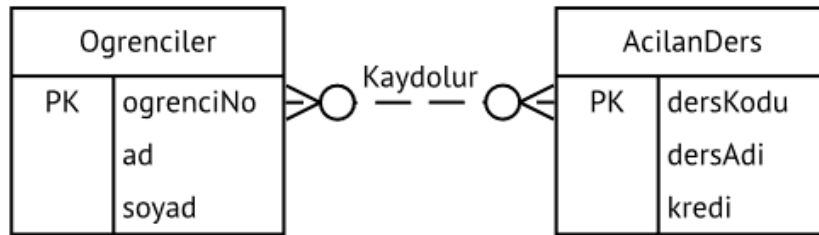
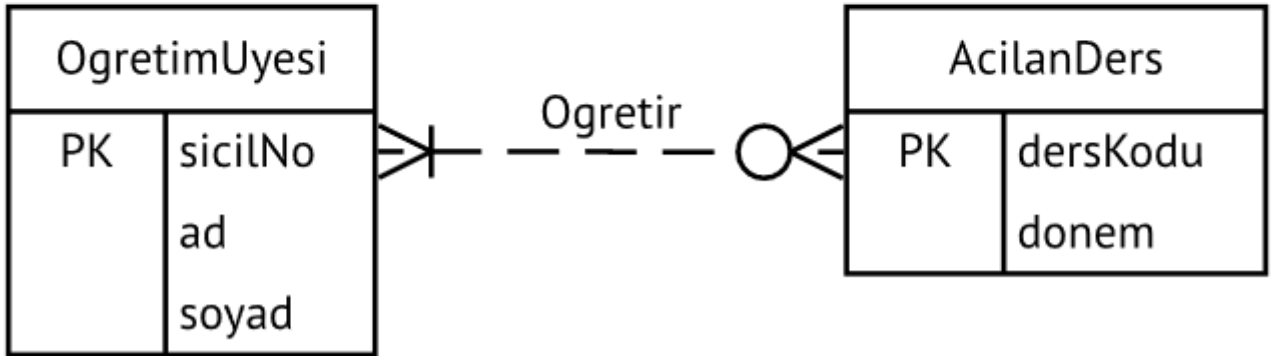
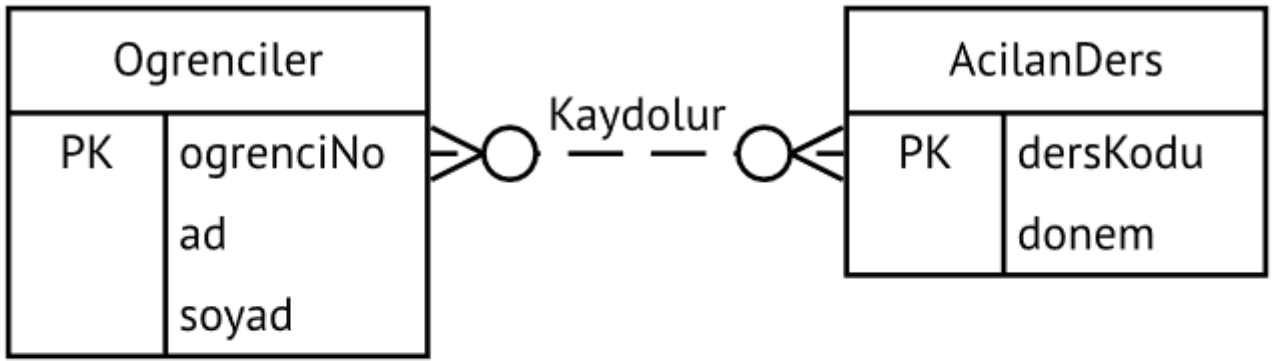
- **1 derslik** sadece **1 binada** bulunabilir.
- **1 binada çok** sayıda **derslik** bulunabilir.



- Normal şartlarda tablo içerisindeki kayıt sayısı sınırlanamaz. Bunun için uygulama yazılımları ya da tetikleyiciler (trigger) kullanılabilir.
- Kayıt sayılarının gösterilmesi uygulama yazılımı geliştirilirken çok faydalıdır. (Sınıfın açılabilmesi için en az 10 kayıt, en fazla 30 kayıt gereklidir. Bir doktor bir günde en fazla 30 reçete yazabilir...)
- Sayılar iş kurallarına bakılarak belirlenir.

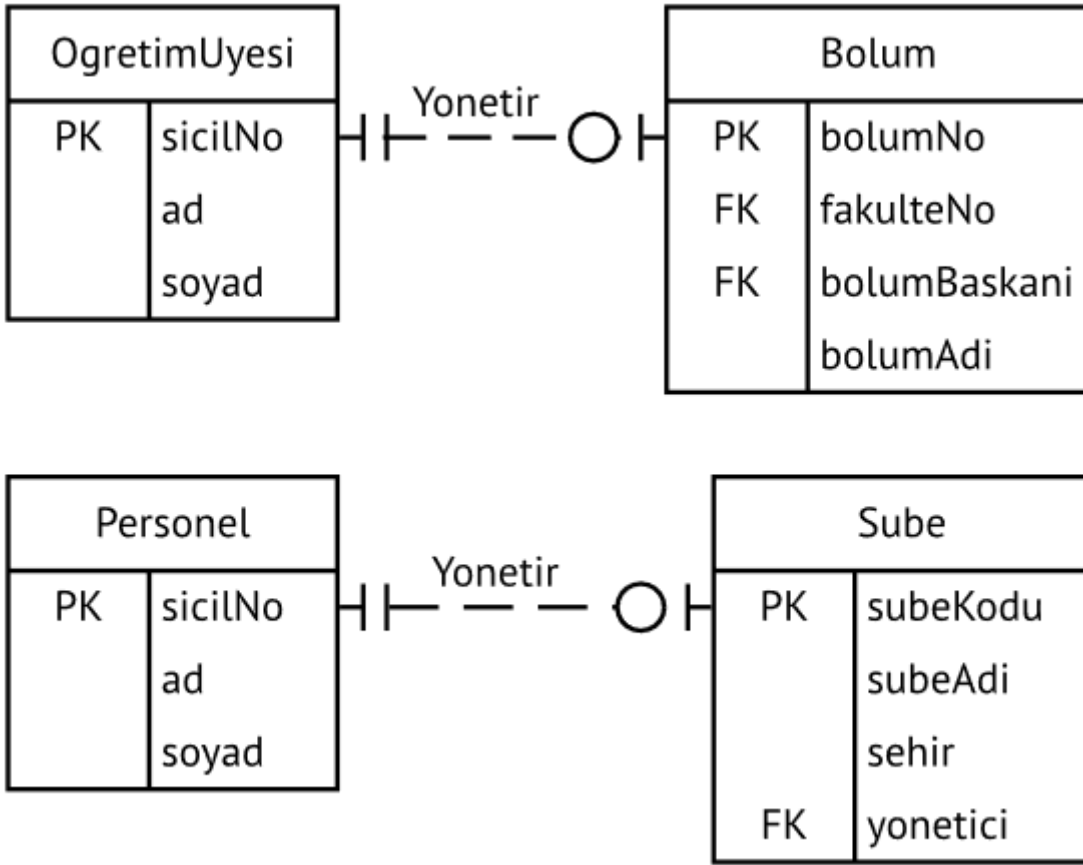
### Çok - Çok Bağıntısı

- **1 öğrenci çok** sayıda **derse** kayıt yaptırabilir
- **1 ders çok** sayıda **öğrenci** tarafından alınabilir.



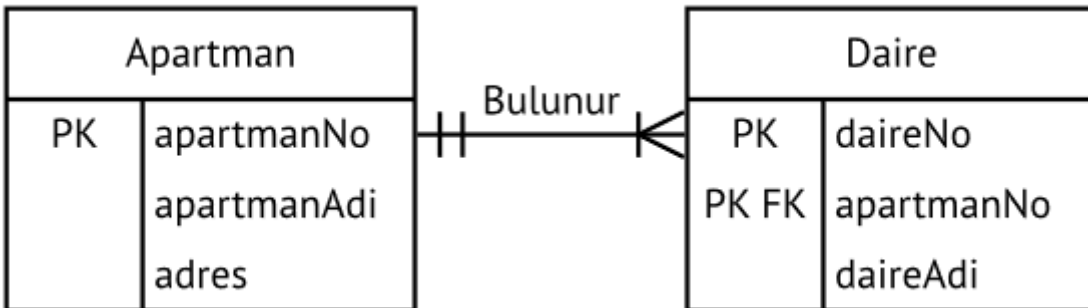
Bir - Bir Bağıntısı

- **1 öğretim üyesi 1 bölüm** yönetebilir
- **1 bölüm 1 öğretim üyesi** tarafından yönetilebilir.



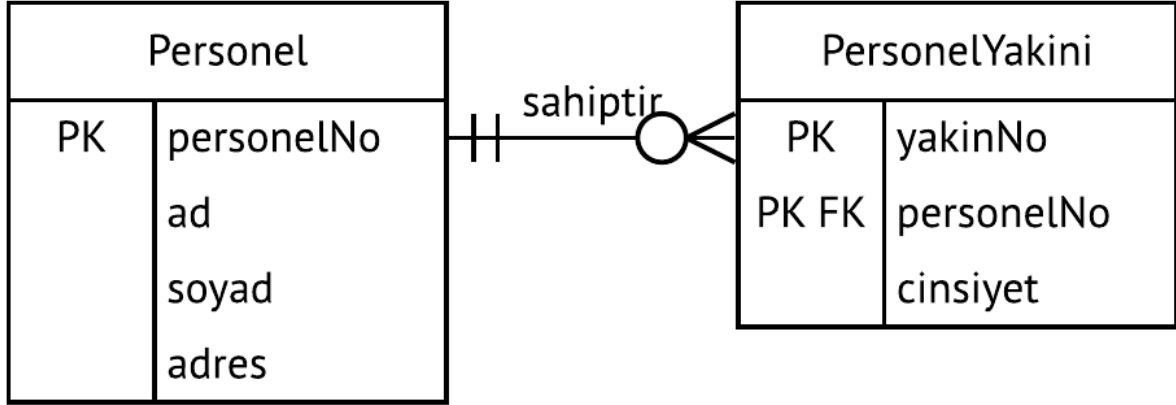
### Var Olma Bağımlılığı (Existence Dependency)

- Bir site yönetim sisteminde Apartman ve Daire varlık kümeleri olsun.
- Böyle bir sistemde «**bir apartmana bağlı olmayan daire olamaz**» kuralı mevcuttur.
- Örneğin Daire varlık kümesine hiçbir apartmana ait olmayan bir dairenin kaydını yapamayız, yapmamalıyız.
- Bu örnekte, Apartman ve Daire arasında var olma bağımlılığı vardır denir.
- Bu durumda Apartman üstün varlık, Daire ise bağımlı varlıktır.

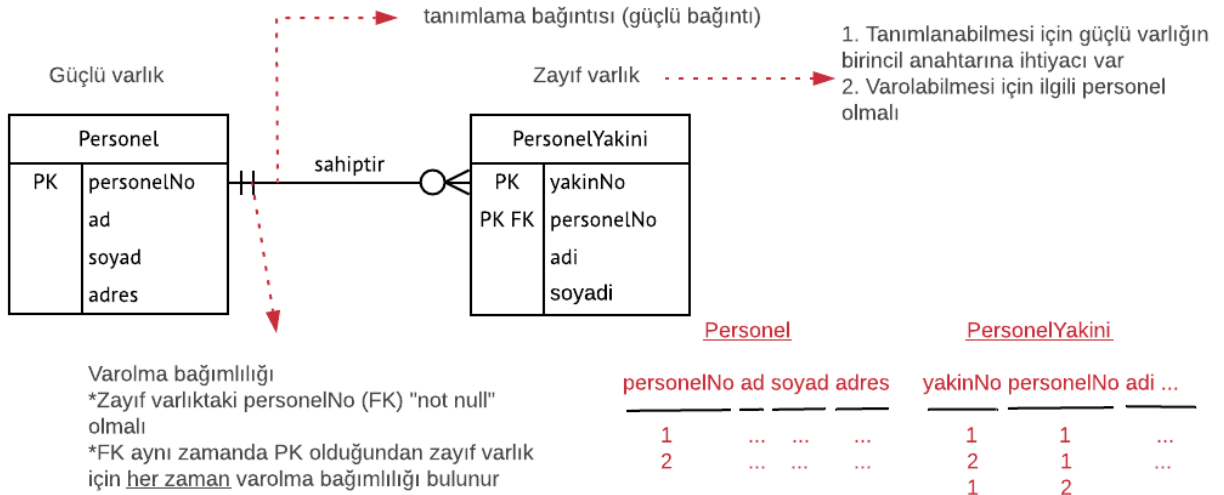


Bir personel bilgi sisteminde Personel ve Bagimli (Personele bagimli olan kişiler. Örneğin çocuk, eş vb.) varlık kümelerini düşünelim. Böyle bir sistemde «**bir personele bağlı olmayan bagimli varlığı olamaz**» kuralı mevcuttur. Örneğin bagimli varlık kümesine hiçbir personele ait olmayan bir çocuğun kaydını yapamayız, yapmamalıyız. Bu örnekte, bagimli ve personel arasında var olma bağımlılığı vardır denir. Bu durumda personel **üstün varlık**, bagimli ise **bağımlı varlıktır**.



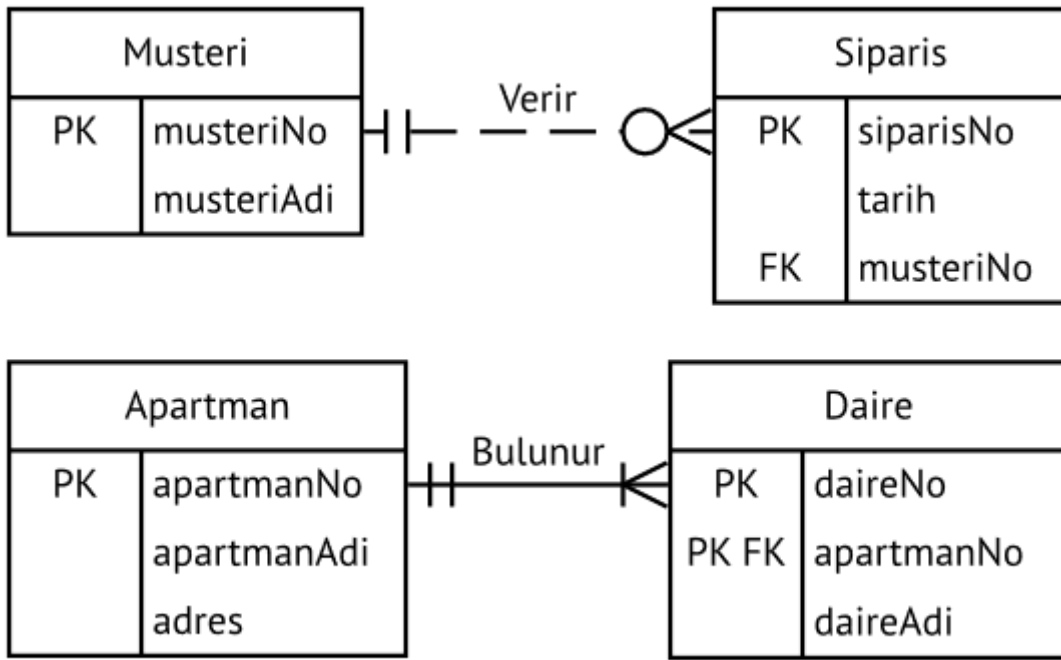


- Var olma Bağımlılığı, Tanımlama Bağıntısı



## Zayıf/Güçlü Bağıntılar

- Bağıntı kurulan varlığın birincil anahtarı içerisinde, bağıntı kuran varlığın birincil anahtar bilgisi yer almıyorsa "iki varlık arasında zayıf bağıntı vardır" denir.
- Bağıntı kurulan varlığın birincil anahtarı içerisinde, bağıntı kuran varlığın birincil anahtarı yer alıyorsa "iki varlık arasında güçlü bağıntı vardır" denir.

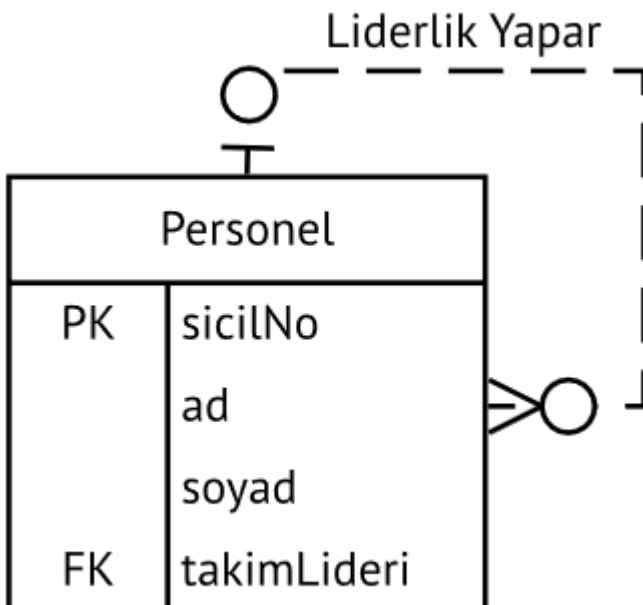


### Bağıntı Dereceleri

- Tekli (Unary) Bağntı
- İkili (Binary) Bağntı
- Üçlü (Ternary) Bağntı

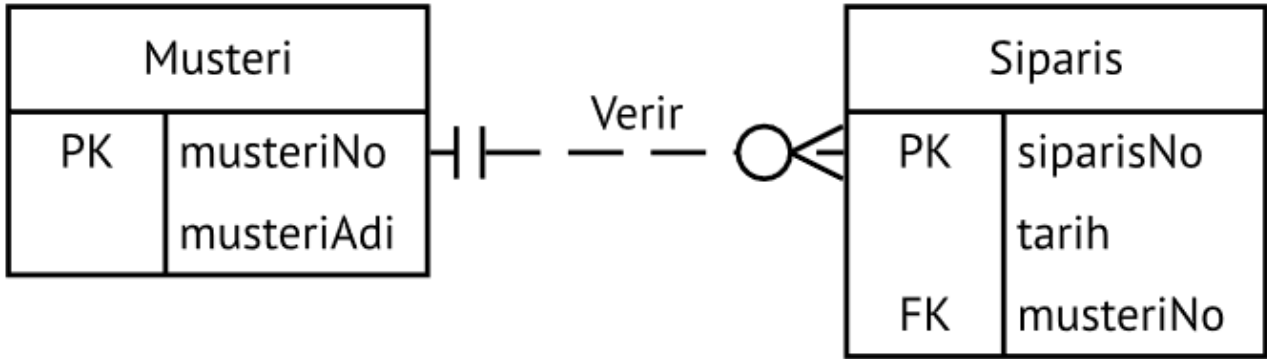
### Tekli Bağntı

- **Tekli (Unary) Bağntı:** Bir varlık kendisi ile bağıntılı (ilişkili) ise bu tür bir bağıntıya tekli bağıntı adı verilir.
- Örneğin, Personel tablosu içerisindeki bir personel, sıfır veya daha fazla personelin aynı zamanda yöneticisidir. Bir personelin sıfır ya da bir yöneticisi olmalıdır.



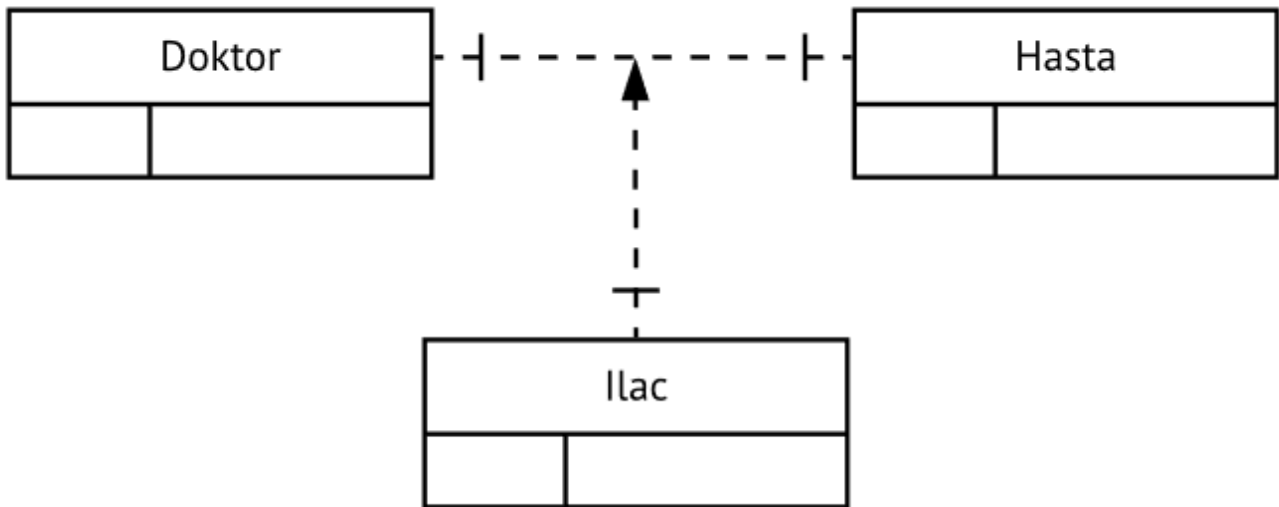
## İkili Bağıntı

- **İkili (Binary) Bağıntı:** İki varlığın bağıntısına (ilişkisine) ikili bağıntı denir.



## Üçlü Bağıntı

- **Üçlü (Ternary) Bağıntı:** Aynı anda 3 varlık birbirine bağlanıyorsa, bu tür bağıntıya üçlü bağıntı adı verilir.
- Kavramsal tasarımda her ne kadar 3 varlık mevcut ise de bunu gerçekleştirebilmek için 4. bir varlığa gereksinim duyulur.



## Varlık Bağıntı Modeli (VBM/ERM) ile Tasarım

- Veritabanı tasarımı, lineer ya da ardışıl (yazılım geliştirmedeki şelale modeli gibi) olmaktan çok tekrarlı (yazılım geliştirmedeki spiral model ya da iteratif model gibi) bir süreçtir.
- Tekrar fiili, "tekrar tekrar yap" anlamındadır. Tekrarlı bir süreç, süreçlerin ve prosedürlerin tekrarlanması temeline dayanır.
- Bir varlık bağıntı diyagramının geliştirilmesi genellikle aşağıdaki adımları içermektedir.
  - Organizasyonun (kurumun) işlerinin (operasyonlarının) tanımını içeren detaylı bir senaryo (hikaye) oluşturulur. Senaryo özellikle organizasyon içerisindeki rol temsilcilerine danışılarak oluşturulursa çok daha gerçekçi ve etkili olur.
  - Senaryoda geçen işlerin tanımları baz alınarak iş kuralları oluşturulur.

- İş kuralları baz alınarak ana varlıklar ve varlıklar arasındaki bağıntılar oluşturulur.
- İlk varlık bağıntı diyagramı geliştirilir.
- Varlıkları net bir şekilde tanımlayan nitelikleri ve birincil anahtarları oluşturulur.
- Varlık Bağıntı Diyagramı gözden geçirilerek gerekirse yukarıdaki adımlar, istenilen duruma gelinceye kadar tekrarlanır.

## Kaynaklar

- Carlos Coronel, Steven Morris, and Peter Rob, Database Systems: Design, Implementation, and Management, Cengage Learning.