# Sistem Analizi ve Tasarımı

BLM2042 GR.1/2

2025-2026 BAHAR YARIYILI

DR.ÖĞR.ÜYESİ YUNUS EMRE SELÇUK / GÖKSEL BİRİCİK

## Bu Derste

Sistem Tasarımı
- Veri Modelleme
- Veri Yapısı ve Veri Tabanı Tasarımı

Kodlama Teknikleri

Yazılım Kalitesi

Test Teknikleri

Yeni Sisteme Geçiş

Bakım

# Varlık-İlişki Modeli

Entity-Relationship Model

- Kavramsal tasarımda veritabanında tutulacak verilerin daha üst seviyede gösterilmesi için
- Kavramsal tasarım için en çok kullanılan ve en popüler model
- Diğer modeller?
  - Sıradüzensel: Kayıt kütükleri
  - Yarı yapılı veri modeli: XML
  - Nesneye dayalı model: Kavramsal nesneler
  - Yapılanmış bellek: NoSQL, MongoDB, AllegroGraph, …

# Varlık-İlişki Sembolleri

# Varlık

Modelin en temel üyesidir.

Var olan ve benzerlerinden ayırt edilebilen her şey varlıktır.

- Ör: Kitap, öğrenci, araba birer varlıktır.

Modelin içerisinde varlık kümesi dikdörtgen ile gösterilir.

Veritabanı olarak düşünülürse her bir tablo bir varlık kümesidir.

# Nitelik

Varlıkların her bir özelliği bir nitelik olarak ifade edilir.

- Ör: öğrenci adı ve numarası öğrenci varlığının nitelikleridir.

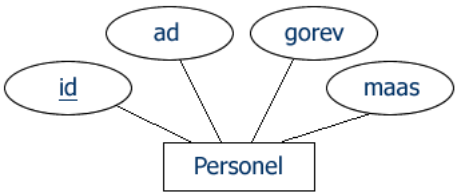
Modelin içerisinde nitelikler oval ile gösterilir ve içerisine niteliğin ismi yazılır.

Veritabanı olarak düşünülürse tablonun her bir sütunu bir niteliği gösterir.

# Anahtar Nitelik

Bir niteliğin değeri her bir varlık için farklıysa bu nitelik anahtar nitelik olarak belirlenir.

Şemada niteliğin altı çizilerek gösterilir.




Personel Varlık Kümesi

# İlişki

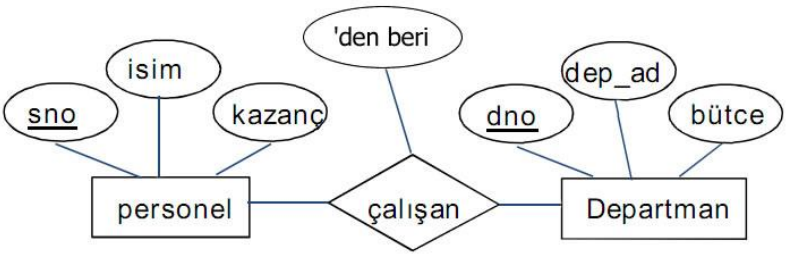
Farklı varlıklar arasındaki ilişkileri ifade eder.

- Ör: öğrenci ve dersler ayrı varlık kümeleridir ama öğrenciler ders almak zorunda olduğu için iki varlık arasında ders alma ilişkisi vardır.

Model içerisinde ilişkiler baklava dilimi ile gösterilir ve içerisine ilişkinin adı yazılır.

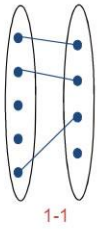
Tablolar arasında kullanılan ilişkiler 1-1, 1-n, n-1, n-m ile gösterilir.

# Varlık-İlişki Örneği



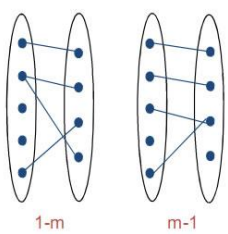

# İlişki Tipleri – Birebir İlişki

- **Bire-bir İlişki :** Herhangi bir varlık kümesindeki her varlık diğer varlık kümesinin en çok bir varlığı ile ilintilidir.




# İlişki Tipleri – Bire-çok İlişki

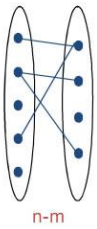
- **Bire-çok ilişki :** İlk kümedeki her varlık diğer kümenin en çok bir varlığına ilintidir.

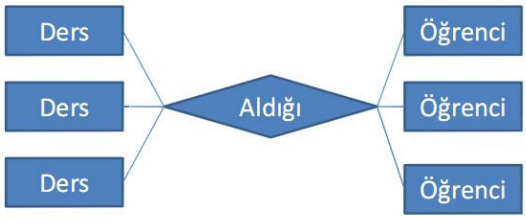




# İlişki Tipleri – Çoka-çok İlişki

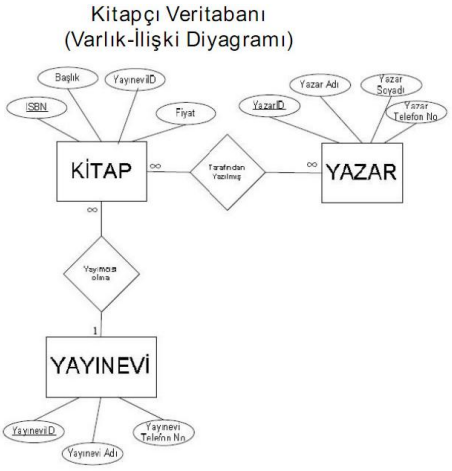
- **Çoka-çok ilişki :** Bir ilişkide herhangi bir kümede bulunan varlıklardan herbiri diğer kümede bulunan birçok varlıkla ilintilidir.




Birincil anahtarları A ve B tablolarının yabancı anahtarlarından oluşan, bağlantı tablosu diye adlandırılan üçüncü bir tablo tanımlayarak oluşturulur.

# Örnek Varlık-İlişki Diyagramı




# Veri Modelinin Gerçeklenmesi

Varlık kümeleri tablolara dönüştürülür.

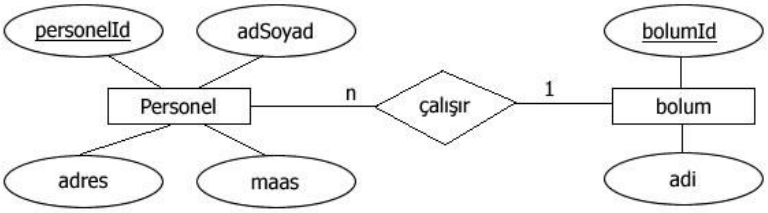
Nitelikler tablonun sütunlarına dönüştürülür.

Modelde oluşturulan ilişkilerin durumuna göre tabloların ilişkileri ve doğal olarak da anahtar sütunları belirlenir.

1-1 İlişkide bir varlık kümesinin birincil anahtarı diğer varlık kümesinin yabancı anahtarı olarak belirlenir. Hangisinin birincil hangisinin yabancı anahtar olacağına tablonun içereceği bilgilere göre karar verilir.

1-n İlişkinin n tarafındaki tabloya 1 tarafındaki tablonun birincil anahtar sütunu yabancı anahtar olarak eklenir.

# Örnek




Primary Key

Foreign Key

**Personel** (personelId, adSoyad, adres, maas, bolumId)

**Bolum** (bolumId, adi)

Primary Key

# Çoka-Çok İlişkilerin Gerçeklenmesi

Varlık kümeleri tablolara dönüştürülür.

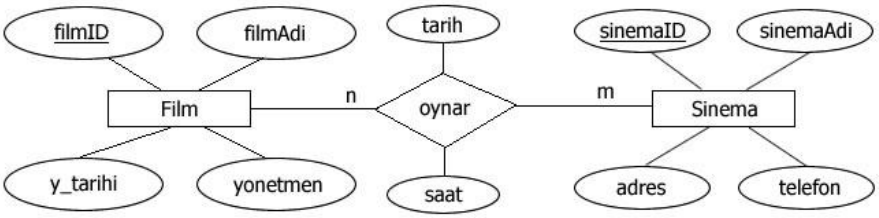
Oluşturulan ilişki isminde tablo oluşturulur.

Nitelikler tabloların sütunlarına dönüştürülür. Tanımlayıcı nitelikler ilişkiden oluşturulan tabloya sütun olarak eklenir.

İlişkiyi oluşturan tabloların birincil anahtarları ilişkiyi oluşturan tabloya yabancı anahtar olarak eklenir.

İlişkiden oluşturulan tablonun birincil anahtarı oluşturulan yabancı anahtarların birleşiminden oluşur. Eğer, bu şekilde oluşturulan birincil anahtar ihtiyaçlara cevap vermiyorsa yeni bir sütun eklenerek birincil anahtar yapılır.

# Örnek




**Film** (filmID, filmAdi, y_tarihi, yonetmen)

**Sinema** (sinemaID, sinemaAdi, adres, telefon)

**Oynar** (oynarID, filmID, sinemaID, tarih, saat)

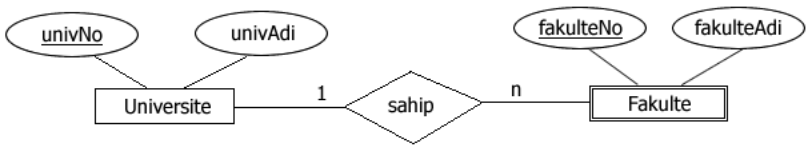
# Zayıf Varlık Kümeleri

Mevcutluğu diğer varlık türüne bağlı olan varlık türüdür.

Eğer bir varlık kümesinin niteliklerinin tümü alınsa bile bir anahtar oluşturmuyorsa buna zayıf varlık kümesi denir.

Ör: Üniversite-fakülte ilişkisinde, bir fakülte üniversite olmadan olamayacağı için ve aynı fakülte isminde başka üniversitelerde fakülte olabileceği için fakülte varlık kümesi zayıf varlık kümesidir.

# Zayıf Varlık Kümelerinin Gerçeklenmesi




Zayıf varlık kümeleri çift çizgili dikdörtgen ile gösterilir.

**Universite** (univNo, univAdi)

**Fakulte** (univNo, fakulteNo, fakulteAdi)

# Kodlama

Tasarım tamamlanıp diyagram, algoritma ve sözde kodlar oluşturulduktan sonra yapılır.

Kodlama spesifikasyonlara uymalı, kolay okunabilmeli, düzenlenebilmeli, test edilebilmeli, değiştirilebilmelidir.

Belirlenen kodlama standartlarına uyulmalıdır.

- Açıklama satırları kullanılmalıdır.
- Kod yazım deseni kullanılmalıdır.

Anlamlı isimlendirmeler yapılmalıdır.

# Programlama Dili ve Ortamı

Gereksinimlere uygun olacak dil(ler) fizibilitede seçilip ön tasarım aşamasında kesinleştirilmişti.

Dil Seçiminde dikkat edilmesi gereken faktörler

- Genel uygulama alanı
- Algoritma ve veri yapılarının karmaşıklığı
- Yazılımın kullanılacağı ortam
- Uygulama koşulları
- Personelin bilgi düzeyi
- Yapılacak yatırım miktarı
- Müşteri koşulları

# Yazılım Kalitesi

**Kalite**: işlevsel gereksinimlere, geliştirme standartlarına ve beklenen tüm özelliklere tamamen uygun yazılım

**Kalite Faktörleri**:

- Doğruluk
- Güvenilirlik
- Verimlilik
- Güvenlik
- Kullanışlılık
- Hata bulma kolaylığı
- Esneklik
- Sınama kolaylığı
- Taşınabilirlik
- Tekrar kullanılabilirlik
- Bağlanabilirlik

# Yazılımda Kalitenin Sağlanması

Planlama aşamasında kalite kontrolü yöntem ve araçları belirlenmeli

Geliştirme sürecinde durak noktalarında yapılanlar gözden geçirilmeli

Kaynak program sınanmalı

Gerçekleştirilmiş projenin gereksinimleri karşılaması gözden geçirilmeli

Bunlar plan dahilinde yapılmalı:

- Yazılım inceleme planı
- Kaynak programı sınama planı
- Kabul muayene planı

# Yazılımın Test Edilmesi

Hataları bulmak için yapılır.

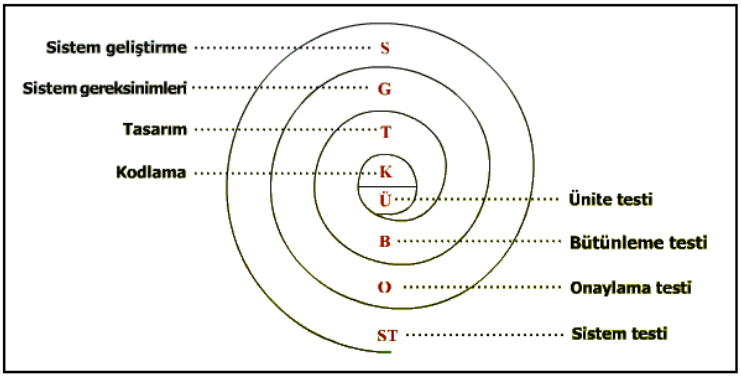
**Fonksiyonel Test:** Girdi-işlem-sonuç üçlüsünün doğruluğu test edilir. Uç değer analizi de yapılır.

**Performans Testi:** Yanıt süreleri, iç-dış bellek kullanımları, iletişim hızları vs. ölçülür. Darboğazlar belirlenir.

**Dayanıklılık (Yük) Testi:** Aşırı yüklenme, iletişim darboğazı, kullanıcı yüklenmesi gibi durumlarda sistemin tepkisi ölçülür.

**Yapısal Test:** Sistemin iç işletimi sınanır. Alt programların mantıksal çalışma yolları denetlenir.

# Test Adımları




# Test Adımları

**Ünite Testi:** Her modüle ayrı uygulanarak kodun doğruluğu test edilir.

**Bütünleme Testi:** Modülleri bağlayarak sistemin oluşturulması sırasında yapılır. Bütün olarak ya da arttırmalı yapılabilir.

**Onaylama Testi:** Gereksinimleri karşılama derecesi test edilir.

**Sistem Testi:** Sistemin bütün öğeleri hep birlikte test edilir. Hatalar için düzeltme, güvenlik, dayanıklılık, performans testleri yapılır.

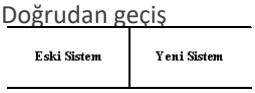
# Yeni Sisteme Geçiş

Geçiş Adımları:

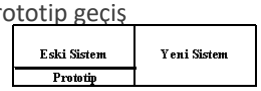
- Ön tasarımda belirlenen donanımın kurulumu
- Ağ yapısının oluşturulması
- Kayıtların yeni sisteme aktarılması
- Eğitim
  - Sistemle ilgilenecekler
  - Uç kullanıcılar
  - Üst yönetim
- Devreye alma

# Geçiş Yöntemleri

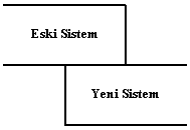
Doğrudan geçiş



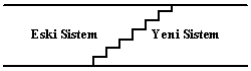

Prototip geçiş



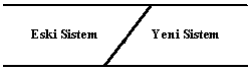

Paralel geçiş




Dağıtılmış geçiş




Dereceli geçiş



# Bakım

Yazılım yaşayan bir süreçtir, evrimleşir.

Kullanıma geçişten sonra yazılımdaki değişikliklere bakım denir.

Zaman içinde ihtiyaçlar değişir. → İyileştirici bakım

Altyapı değişimi olabilir. → Uyarlayıcı bakım

Testlerde fark edilmeyen hatalar olabilir. →Düzeltici bakım

# Gelecek Ders

Vaka Çalışması