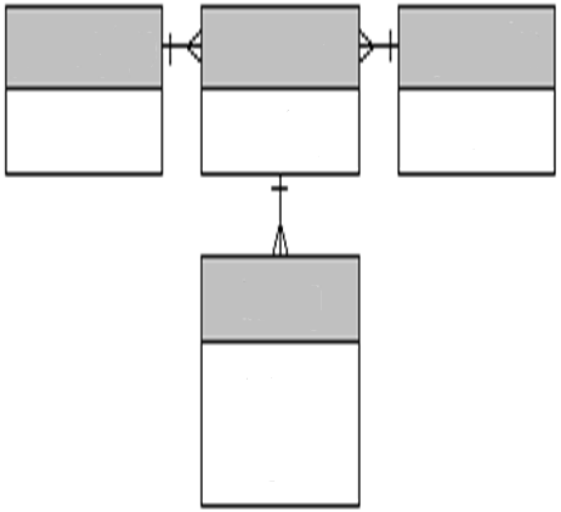
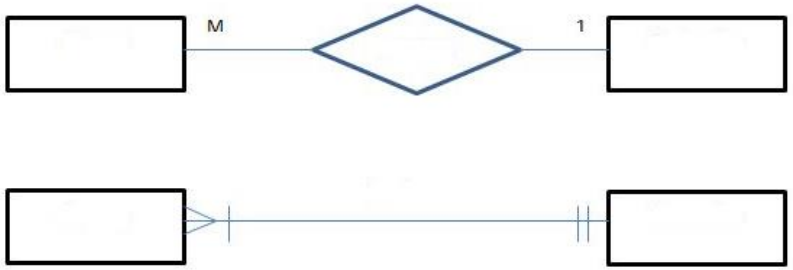
Veri Tabanı Yönetim Sistemleri



Hafta 3 – Varlık Bağıntı Modeli

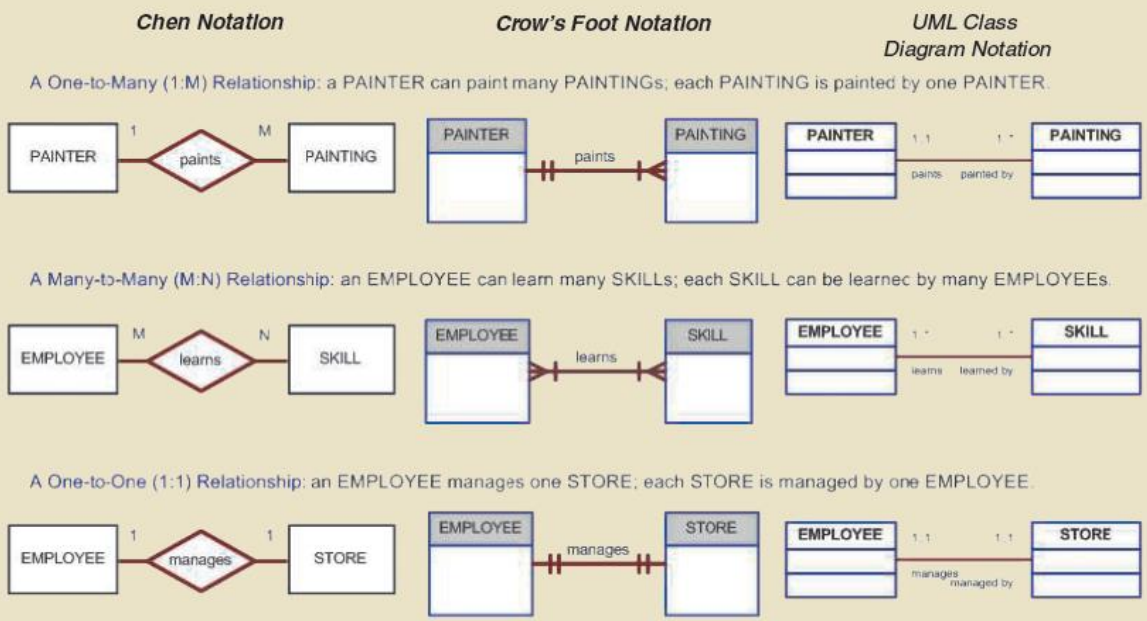


Varlık Bağıntı

- Varlık Bağıntı Modelleri (VBM – Entity Relationship Model (ERM)) , Varlık Bağıntı Diyagramlarının (VBD – Entity Relationship Diagram (ERD)) temelini oluşturur.
- Varlık İlişki Diyagramı, veritabanının kavramsal modellemesini sağlayan yazılım ve donanımdan tamamen bağımsız bir temsildir.
- VBD, son kullanıcı tarafından görüntülendiği şekliyle kavramsal veritabanını temsil eder
- VBD, veri tabanının temel bileşenleri olan ***Varlık, Nitelik (özellik), varlıklar arası Bağıntılar (Relationship)*** ve kısıtlardan meydana gelen çizelgelerdir.

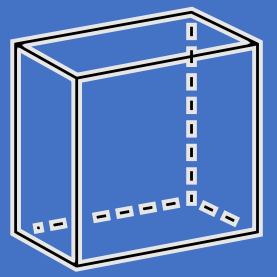
Varlık Bağıntı

FIGURE 2.3 THE ER MODEL NOTATIONS



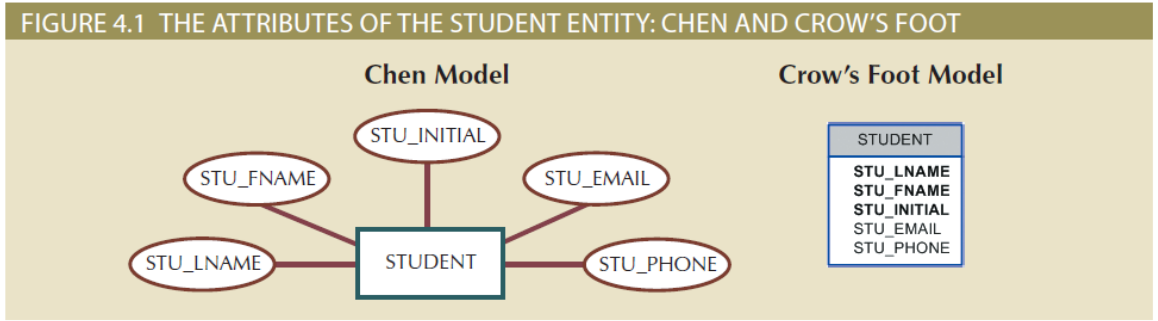
- VB modeline bakarak veri tabanının tasarımını gerçekleştirmek daha kolaydır.
- Chen notasyonu kavramsal modellemeyi ön plana çıkartır.
- Crow's Foot notasyonu ise daha çok uygulama (implementasyon) yönelimli yaklaşımı ön plana çıkartır.
- UML notasyonu, hem kavramsal hem de uygulama (implementasyon) modelleme yaklaşımı için kullanılabilir.

Varlık (Entity)



Varlık

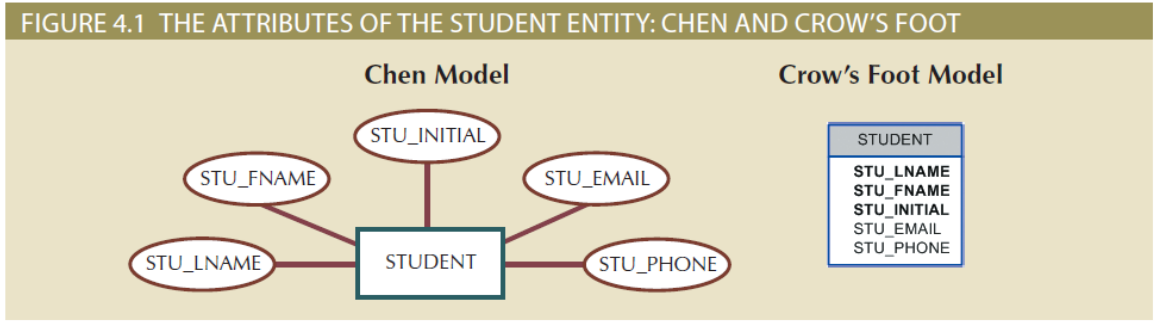
- Son kullanıcının ilgilendiği nesnedir.
- VBM'deki bir varlık, ilişkisel ortamda bir tabloya karşılık gelir (bir satıra değil).
- Chen, Crow's Foot ve UML gösterimlerinde bir varlık, varlığın adını içeren bir dikdörtgenle temsil edilir.
- Varlık adı, genellikle tamamı büyük harflerle yazılır (isim).



Nitelik (Attribute)

Nitelik

- Nitelikler, varlıkların özellikleridir (karakteristik).
- Orijinal Chen gösteriminde, nitelikler ovallerle temsil edilir ve varlık dikdörtgenine bir çizgi ile bağlanır. Her oval, temsil ettiği niteliğin adını içerir.
- Crow's Foot gösteriminde, öznitelikler, varlık dikdörtgeninin altındaki öznitelik kutusuna yazılır.
- Chen gösterimi daha fazla yer kaplamaktadır.

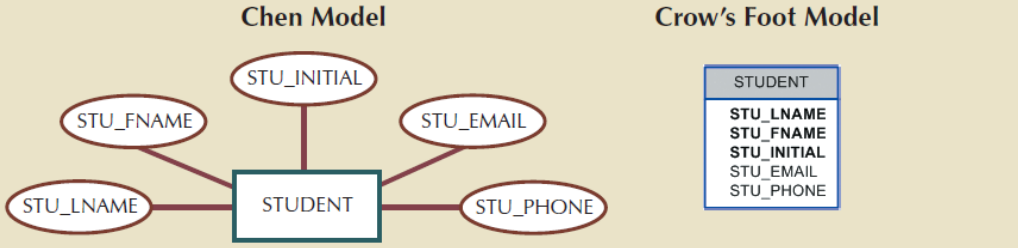


Nitelik (Attribute)

Gerekli ve Opsiyonel Nitelikler

- **Gerekli Nitelik;** Bir değeri olması gereken niteliklerdir.
  - Crow's Foot gösterimindeki kalın karekterli nitelikler, veri girişinin gerekli olduğunu gösterir.
- **Opsiyonel Nitelik;** Bir değer gerektirme zorunluluğu bulunmayan bu nedenle boş bırakılabilir niteliklerdir.

FIGURE 4.1 THE ATTRIBUTES OF THE STUDENT ENTITY: CHEN AND CROW'S FOOT



Nitelik (Attribute)

Alan (Domain)

- Niteliklerin alanı (domain) bulunur.
- Etki alanı, belirli bir nitelik için olası değerler kümesidir.

  Örn:

  - Bir not ortalaması (GPA) niteliği için etki alanı (0,4) yazılabilir. Çünkü mümkün olan en düşük GPA değeri 0'dır ve mümkün olan en yüksek değer 4'tür.
  - Bir cinsiyet niteliği alanı yalnızca iki olasılıktan oluşabilir: E veya K (veya başka bir eşdeğer kod).
  - Bir şirketin işe alma tarihi niteliğinin alanı belirli bir tarih aralığına uyan tüm tarihlerden oluşur (Şirket başlangıç tarihinden bugünkü geçerli tarihe kadar).
- Nitelikler bir alanı paylaşabilir. Örneğin, PROFESÖR ve ÖĞRENCİ varlıklarının her birinin ADRES adlı bir niteliği olabilir ve bu nedenle bir alanı paylaşabilir.

Nitelik (Attribute)

Identifiers (Primary Keys)

- VBD, tanımlayıcılar kullanır - her varlık örneğini benzersiz şekilde tanımlayan bir veya daha fazla nitelik.
- İlişkisel modelde varlıklar tablolarla eşlenir ve varlık tanımlayıcısı tablonun birincil anahtarı (PK) olarak eşlenir.
- Tanımlayıcıların altı ERD'de çizilmiştir.
- Anahtar nitelikler sonraki haftalarda değineceğimiz İlişkisel Şema larda da altı çizili olarak gösterilirler.

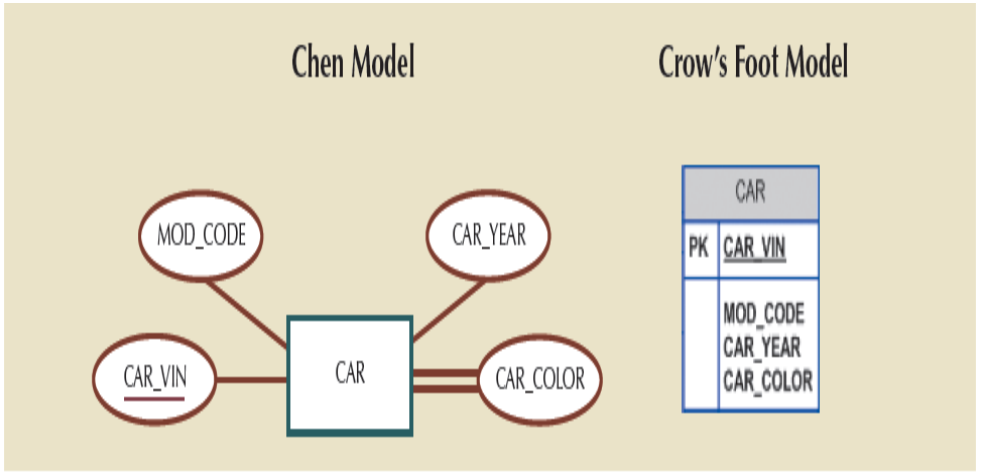


TABLE NAME (**KEY_ATTRIBUTE 1**, ATTRIBUTE 2, ATTRIBUTE 3, … ATTRIBUTE K)

CAR (**CAR_VIN**, MOD_CODE, CAR_YEAR, CAR_COLOR)

Nitelik (Attribute)

Birleşik ve Basit Nitelikler

- **Basit Nitelik;** Alt bölümlere ayrılamayan niteliklerdir.
  - Yaş, Ad, Soyad, Cinsiyet vb.

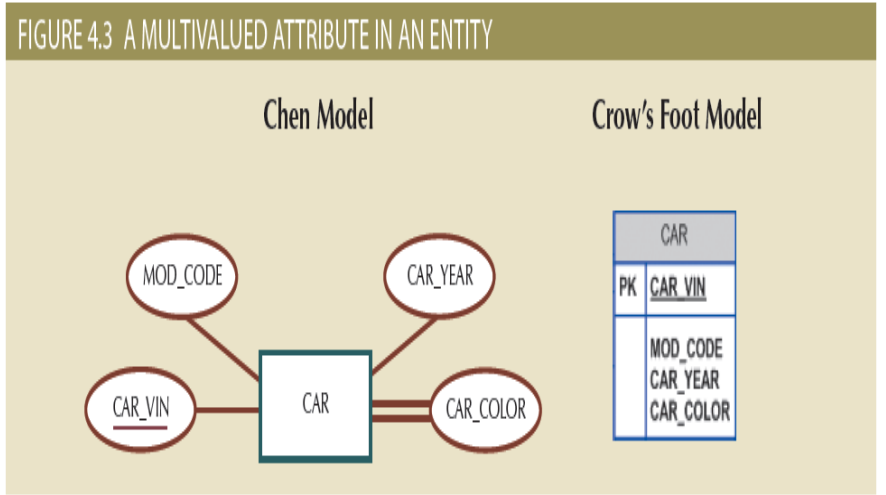
- **Birleşik Nitelik;** Ek nitelikler oluşturabilecek şekilde daha fazla alt bölümlere ayrılabilen bileşik niteliklerdir.
  - ADRES niteliği; cadde, şehir ve posta kodu olarak alt bölümlere ayrılabilir.
  - Detaylı sorguları kolaylaştırmak için basit özellikli alanlara dönüştürülmeleri gerekir.
  - Veritabanı tasarımcısı her zaman birleşik niteliklere dikkat etmelidir. İş kurallarında birleşik nitelikler kullanılması yaygındır ve kullanıcılar genellikle ortamlarındaki varlıkları bileşik nitelikler kullanarak tanımlar. Tasarımcı, birleşik özelliklerin farkına varmalı ve basit özelliklere ayırmanın doğru yolunu belirlemelidir.

Nitelik (Attribute)

Tek-Değerli ve Çok-Değerli Nitelikler

- **Tek-Değerli (Single-Valued) Nitelik;** Yalnızca tek bir değere sahip olabilen nitelik.
  - Bir kişinin yalnızca bir T.C Kimlik numarası olabilir.
- **Çok-Değerli (Multi-Valued) Nitelik;** Birçok değere sahip olabilen nitelik.
  - Bir kişinin birkaç üniversite derecesi olabilir,
  - Bir arabanın rengi tavan, gövde ve döşeme için birçok renge bölünebilir.

  - Chen gösteriminde, çok değerli nitelikler, niteliği varlığa bağlayan çift çizgi ile gösterilir. Crow's Foot gösterimi, çok değerli nitelikleri tanımlamaz.



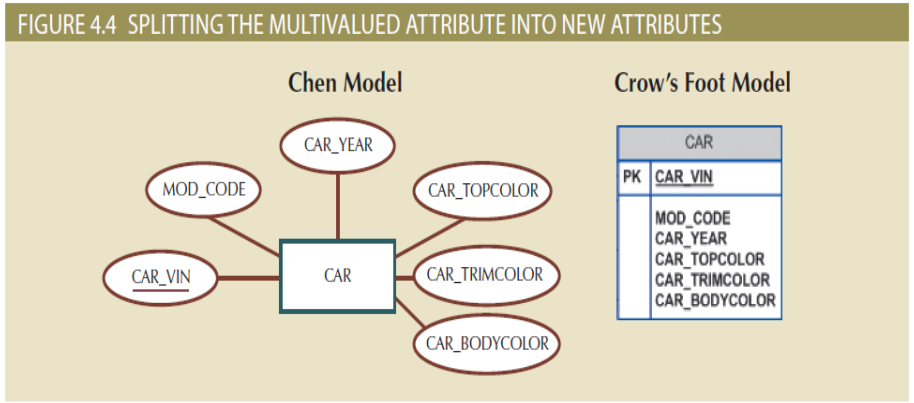
Çok-Değerli Niteliklerin Uygulanması

- Kavramsal model M:N bağıntıları ve çok-değerli nitelikleri uygulayabilmesine rağmen, bunları İlişkisel Veri Tabanı Yönetim Sistemi (RDBMS) üzerinde uygulamamanız gerekir.
- İlişkisel tabloda her bir satır ve sütun kesişimi tek bir değeri temsil eder.
- Bu nedenle eğer çok-değerli nitelikler mevcut ise tasarımcının ne gibi çözümleri bulunabilir?

Çok-Değerli Niteliklerin Uygulanması

- Kavramsal model M:N bağıntıları ve çok-değerli nitelikleri uygulayabilmesine rağmen, bunları İlişkisel Veri Tabanı Yönetim Sistemi (RDBMS) üzerinde uygulamamanız gerekir.
- İlişkisel tabloda her bir satır ve sütun kesişimi tek bir değeri temsil eder.
- Bu nedenle eğer çok-değerli nitelikler mevcut ise tasarımcı genellikle iki muhtemel eylem planından birine karar vermelidir:
  - 1- Orijinal varlık içinde, orijinal çok-değerli niteliğin her bileşeni için bir tane olmak üzere birkaç yeni nitelik oluşturmak.
  - 2- Orijinal çok-değerli niteliğin bileşenlerinden oluşan yeni bir varlık oluşturmak.

FIGURE 4.4 SPLITTING THE MULTIVALUED ATTRIBUTE INTO NEW ATTRIBUTES

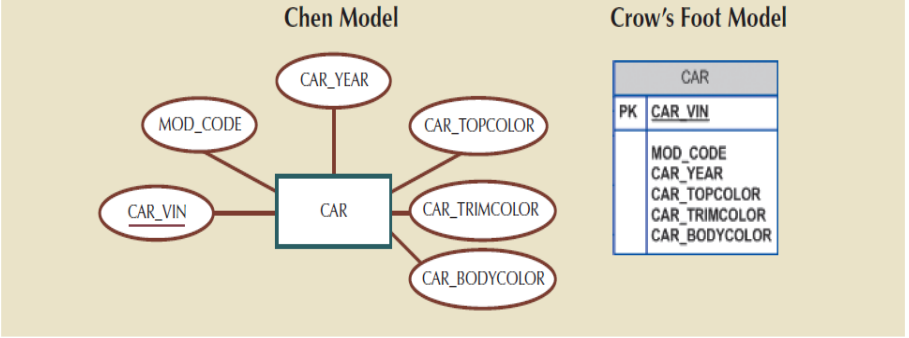


Çok-Değerli Niteliklerin Uygulanması

1- Orijinal varlık içinde, orijinal çok-değerli niteliğin her bileşeni için bir tane olmak üzere birkaç yeni nitelik oluşturmak.

- CAR(ARAÇ) varlığının CAR_COLOR niteliği; CAR_TOPCOLOR, CAR_BODYCOLOR, ve CAR_TRIMCOLOR, yeni niteliklerine ayrıştırılabilir ve ana varlığa bağlanabilir.
- Bu uygulamanın problemleri neler olabilir?

FIGURE 4.4 SPLITTING THE MULTIVALUED ATTRIBUTE INTO NEW ATTRIBUTES



Çok-Değerli Niteliklerin Uygulanması

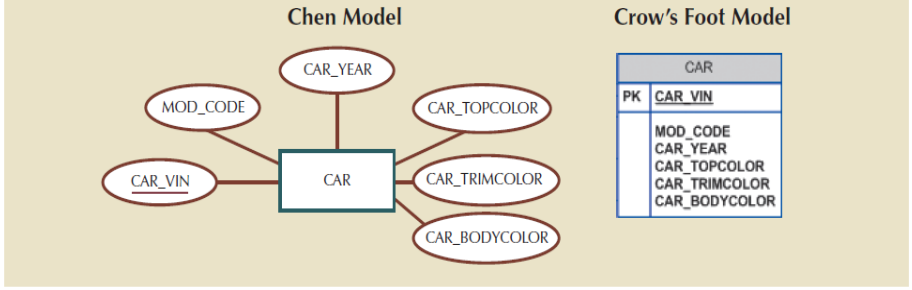
1- Orijinal varlık içinde, orijinal çok-değerli niteliğin her bileşeni için bir tane olmak üzere birkaç yeni nitelik oluşturmak.

- CAR(ARAÇ) varlığının CAR_COLOR niteliği; CAR_TOPCOLOR, CAR_BODYCOLOR, ve CAR_TRIMCOLOR, yeni niteliklerine ayrıştırılabilir ve ana varlığa bağlanabilir.
- Bu uygulamanın problemleri neler olabilir?

-> Yalnızca varlığın her örneğinin çok-değerli nitelik için aynı sayıda değere sahip olması ve hiçbir örneğin daha fazla değere sahip olmaması durumunda kabul edilebilir. Bununla birlikte, ortamdaki yeni değişikliklerin bir örneğin öncekinden daha fazla değere sahip olmayacağı da garanti olmayabilir.

Örneğin, bazı araçlar için LOGO_COLOR(logo rengi) gibi ek renk bileşenleri eklenirse, tablo yapısı yeni renk bölümünü içerecek şekilde değiştirilmelidir. Bu durumda, bu tür renk bölümlerine sahip olmayan arabalar, mevcut olmayan bileşenler için boş değerler oluşturur veya bu bölümler için renk girişleri, "uygulanamaz" ı belirtmek için N/A (not applicable) olarak girilir.

FIGURE 4.4 SPLITTING THE MULTIVALUED ATTRIBUTE INTO NEW ATTRIBUTES



Çok-Değerli Niteliklerin Uygulanması

2- Orijinal çok-değerli niteliğin bileşenlerinden oluşan yeni bir varlık oluşturmak.

- Bu yeni varlık, tasarımcının arabanın farklı bölümleri için renk tanımlamasına olanak tanır. Yeni CAR_COLOR varlığı orijinal CAR varlığına 1:M ilişki ile bağlanır.
- Tablo yapısını değiştirmek zorunda kalmadan gerektiği kadar renk atayabilirsiniz.
- Bu yöntem çok-değerli nitelikler için tercih edilen yöntemdir.
- Orijinal varlık ile 1:M ilişkiye sahip yeni bir varlık oluşturmak birkaç yönden fayda sağlamaktadır: daha esnek, genişletilebilir bir çözümdür ve ilişkisel modelle uyumludur.

TABLE 4.1

COMPONENTS OF THE MULTIVALUED ATTRIBUTE

SECTION	COLOR
Top	White
Body	Blue
Trim	Gold
Interior	Blue

FIGURE 4.5 A NEW ENTITY SET COMPOSED OF A MULTIVALUED ATTRIBUTE'S COMPONENTS

Nitelik (Attribute)

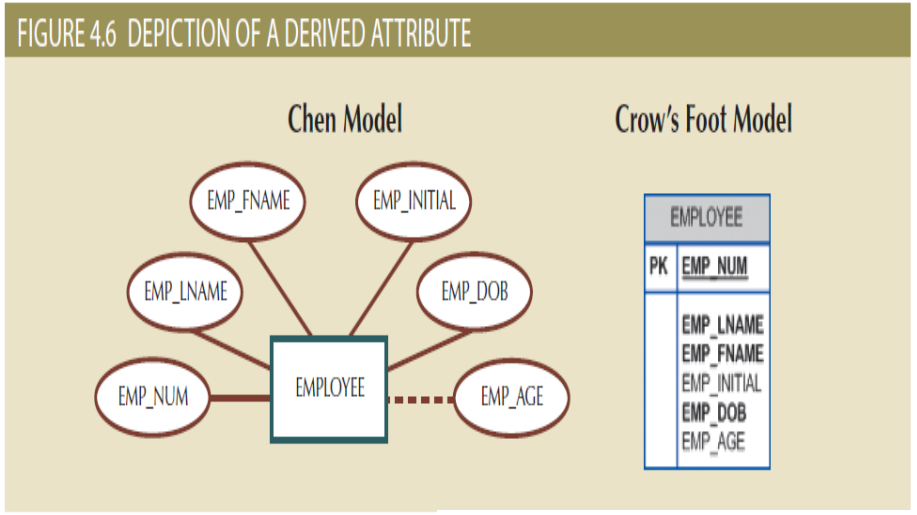
Türetilmiş Nitelikler

- Değeri diğer özellikler kullanılarak oluşturulan özellikler.
- Türetilmiş (Derived) niteliklere bazen hesaplanmış (computed) nitelikler de denir.
  - Yaş değeri güncel tarih ile doğum tarihi arasındaki farktan hesaplanabilir.
  - Aynı şekilde yaş ortalamaları alınarak Kişilerin yaş ortalaması hesaplanabilir.

```
SELECT AVG(AGE(dogumTarihi)) FROM Kisiler
```

- Chen gösterimi: özniteliği ve varlığı birbirine bağlayan kesikli çizgi.
- Crow's Foot gösterimi türetilmiş niteliği diğer özelliklerden ayırt etmek için bir yöntem sunmamaktadır.

- Türetilmiş nitelik ihtiyaç anında hesaplanmalı mı yoksa saklanmalı mıdır?



Türetilmiş Nitelikler

Türetilmiş nitelik ihtiyaç anında hesaplanmalı mı yoksa saklanmalı mıdır?

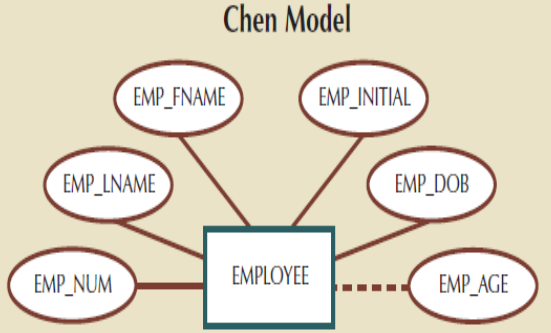
- Saklanması Durumunda:
  - Avantaj: Az işlemci gücü gerekir, veriye daha hızlı erişim, geçmiş bilgisi için kullanılabilir
  - Dezavantaj: Güncel değer için sürekli denetlenmelidir, fazladan yer kaplar
- Hesaplanması Durumunda:
  - Avantaj: Yer tasarrufu sağlar. Her an güncel değer olur.
  - Dezavantaj: Çok işlemci gücü gerekir, veriye daha yavaş erişim olur, sorgular daha karmaşık olur.

TABLE 4.2

ADVANTAGES AND DISADVANTAGES OF STORING DERIVED ATTRIBUTES

	DERIVED ATTRIBUTE	
	STORED	**NOT STORED**
Advantage	Saves CPU processing cycles Saves data access time Data value is readily available Can be used to keep track of historical data	Saves storage space Computation always yields current value
Disadvantage	Requires constant maintenance to ensure derived value is current, especially if any values used in the calculation change	Uses CPU processing cycles Increases data access time Adds coding complexity to queries

FIGURE 4.6 DEPICTION OF A DERIVED ATTRIBUTE



Bağıntı (Relationship)

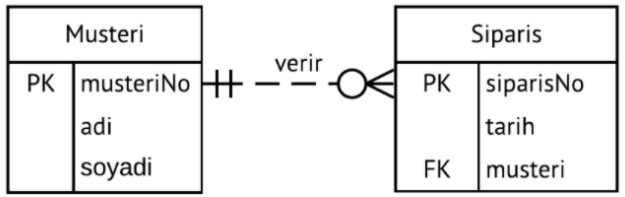
- Bağıntı; varlıklar arasındaki ilişkilerdir.
- Her bağıntı, ilişkiyi açık bir şekilde tanımlayan bir isimle ifade edilir.
- İlişki adı aktif veya pasif bir fiildir.

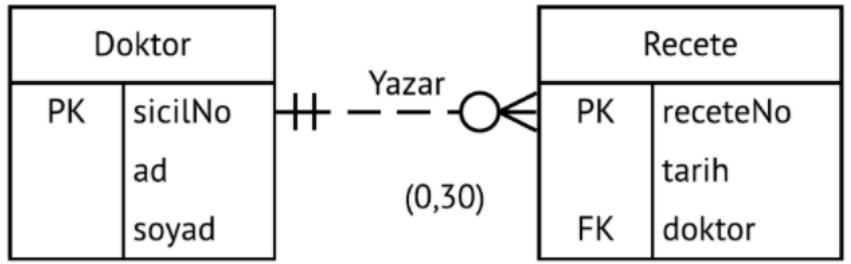
  Örn: Bir ÖĞRENCİ birçok DERS alır. Bir SINIF ta birçok ÖĞRETMEN ders verebilir.
- Varlıklar arasındaki bağıntılar çift yönlü olarak ele alınmalıdır:
  - Bir MÜŞTERİ birçok FATURA oluşturur.
  - Her FATURA bir MÜŞTERİ tarafından oluşturulur.
- Bir-Çok, Çok-Çok ve Bir-Bir Bağıntı

Bağıntı (Relationship)

Bir - Çok Bağıntı 1:M

- Bir varlık örneği bağıntılı varlığın birçok örneği ile ilişkilendirilir.
- 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.

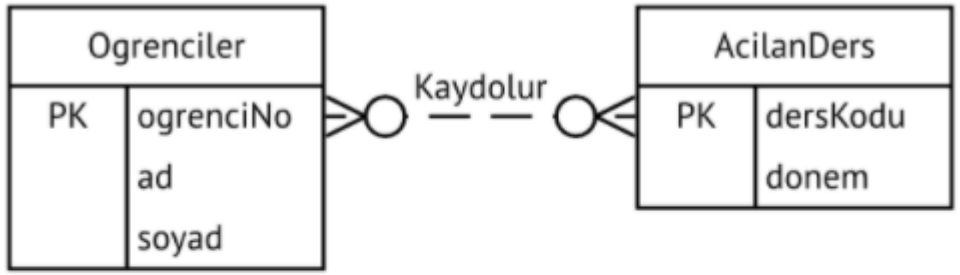


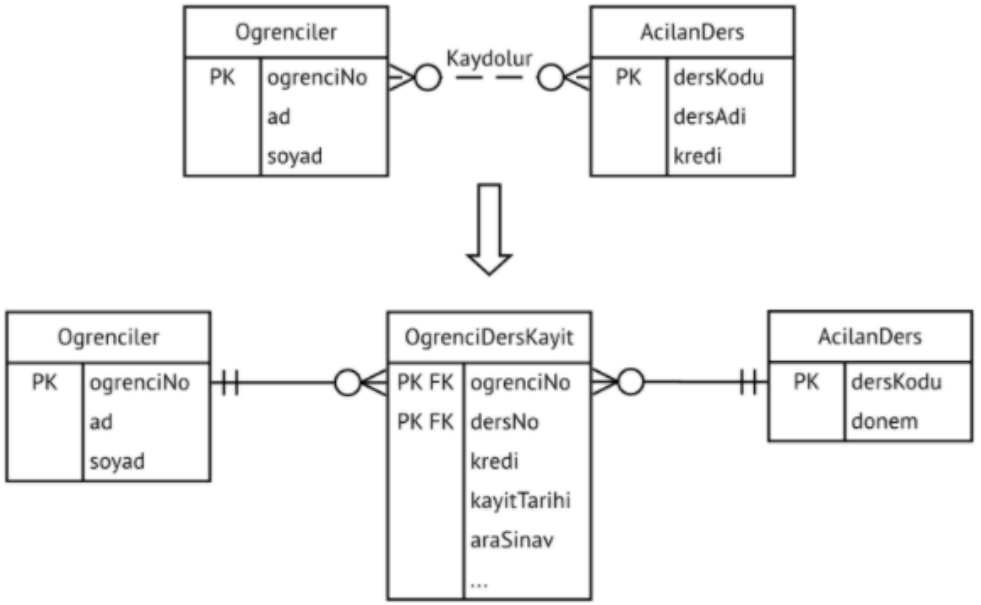


Bağıntı (Relationship)

Çok - Çok Bağıntı M:N

- Bir varlık örneği, ilişkili varlığın birçok örneğiyle ilişkilidir ve ilişkili varlığın bir örneği, ilk varlığın birçok örneğiyle ilişkilidir.
- Ara varlıklar (Örn. ÖğrenciDersKayıt) daha sonra ele alınacaktır.

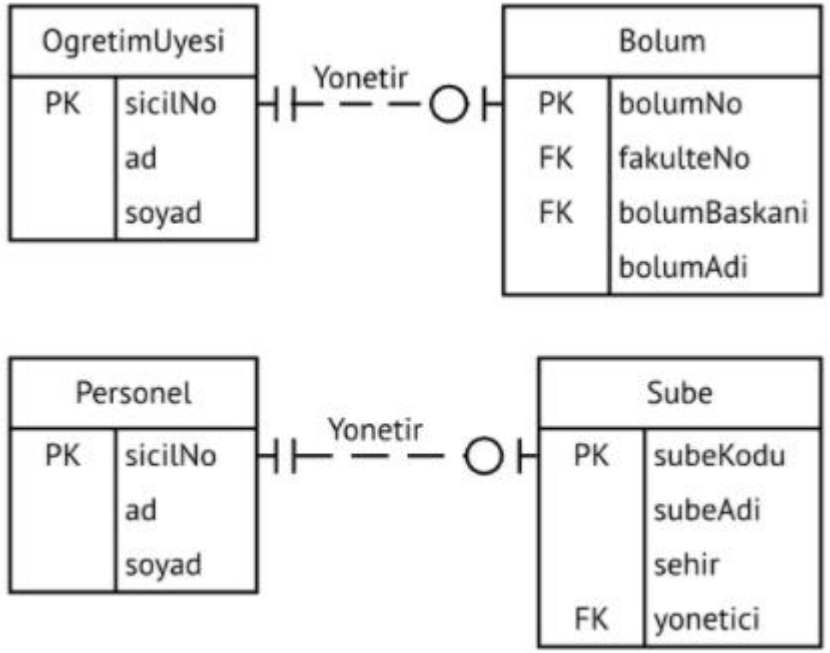




Bağıntı (Relationship)

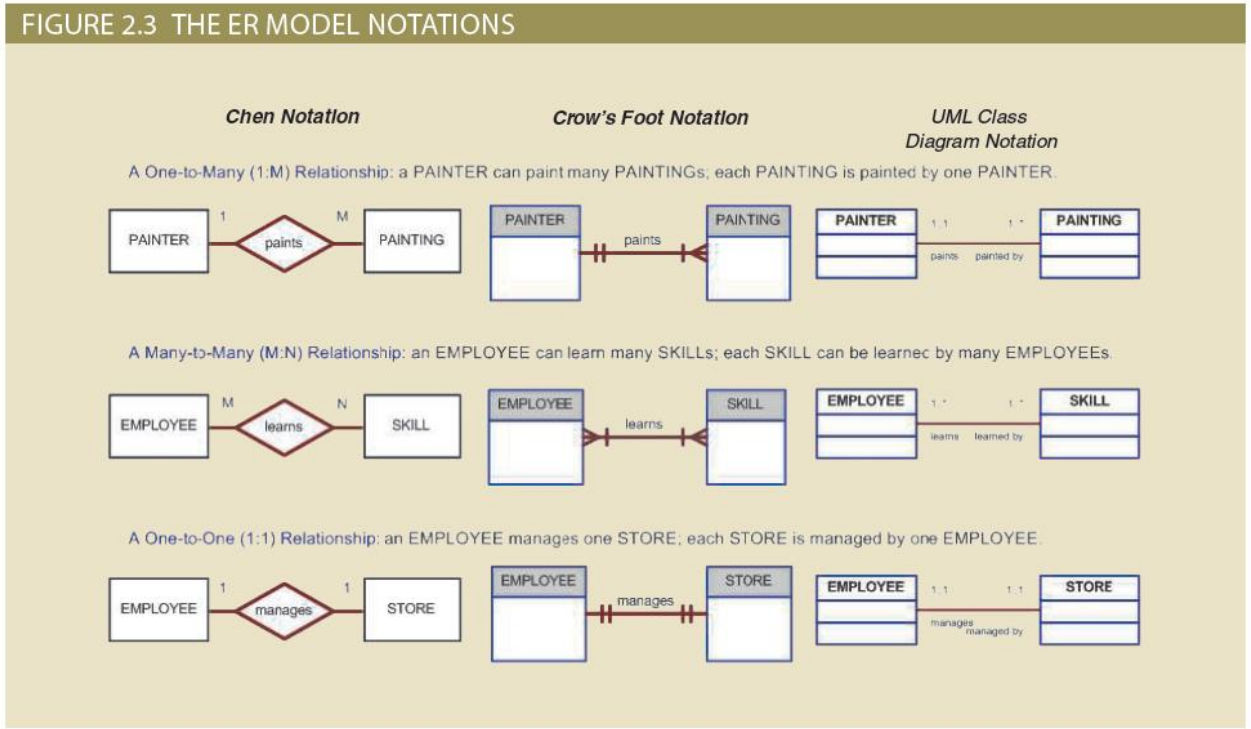
Bir - Bir Bağıntı 1:1

- Bir varlık örneği, ilişkili varlığın yalnızca bir örneği ile ilişkilendirilir.

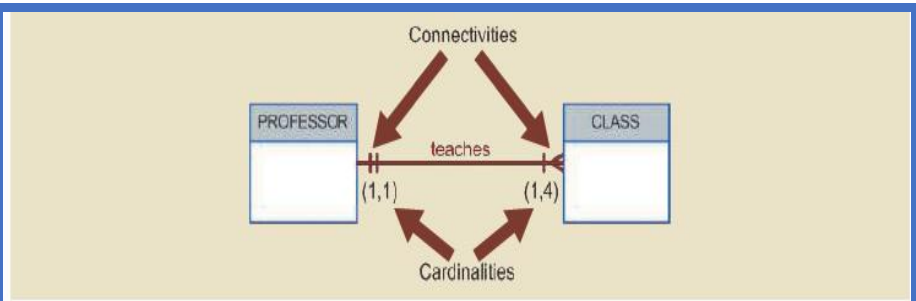


Bağıntı (Relationship)

Farklı İlişki Türlerinin Farklı Diyagramlar Üzerinde Gösterimi



Bağıntı (Relationship)



Kardinalite (Cardinality);

- İlgili varlığın bir oluşumuyla ilişkili minimum ve maksimum varlık sayısını ifade eder.
- VBD'de kardinalite (x, y) formatı kullanılarak varlıkların yanına uygun sayılar yerleştirilerek gösterilir. İlk değer, ilişkili varlıkların minimum sayısını temsil ederken, ikinci değer, ilişkili varlıkların maksimum sayısını temsil eder.
- Minimum ve maksimum varlık sayısını bilmek, uygulama yazılımı düzeyinde çok yararlıdır.

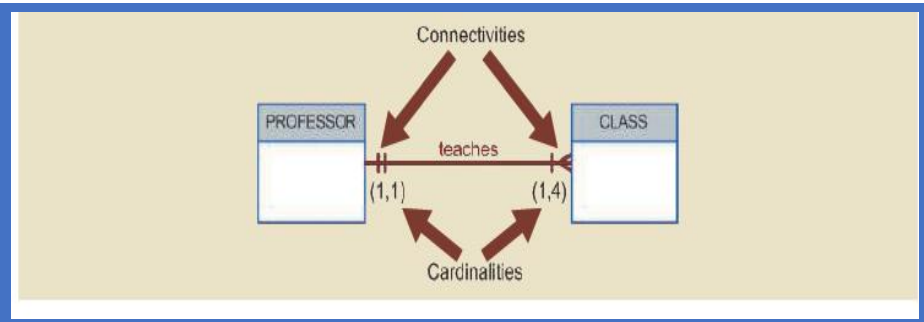
- VTYS kullanırken kardinaliteler genellikle uygulama yazılımı veya tetikleyiciler (trigger) tarafından sağlanır. İleri Seviye SQL'de tetikleyicilerin nasıl oluşturulacağını ve çalıştırılacağını göreceğiz.

TABLE 4.3

CROW'S FOOT SYMBOLS

SYMBOL	CARDINALITY	COMMENT
⟨	(0,N)	Zero or many; the "many" side is optional.
|⟨	(1,N)	One or many; the "many" side is mandatory.
||	(1,1)	One and only one; the "1" side is mandatory.
○|	(0,1)	Zero or one; the "1" side is optional.

Bağıntı (Relationship)



Kardinalite (Cardinality);

- Her bir Profesör en fazla dört sınıfa kadar ders verir; bu, PROFESÖR tablosunun birincil anahtar değerinin, SINIF tablosunda yabancı anahtar değerleri olarak en az bir ve en fazla dört kez bulunduğu anlamına gelir.
- Kardinalite (1, N) olarak yazılsaydı, bir profesörün öğretebileceği sınıfların sayısında üst sınır olmayacaktı.
- Her sınıf bir ve yalnızca bir profesör tarafından verilmektedir. Yani, her bir SINIF varlığı oluşumu, PROFESSOR'da bir ve yalnızca bir varlık oluşumuyla ilişkilidir.
- Bağlantılar ve kardinalite, iş kurallarında belirlenir. İş kuralları VBM'nin bileşenlerini tanımladığından, tüm uygun iş kurallarının tanımlandığından emin olmak, bir veritabanı tasarımcısının işinin önemli bir parçasıdır.
- Chen gösterimi, kardinaliteleri ilgili varlığın yanına yerleştirir. Crow's Foot ve UML diyagramları, kardinaliteleri uygulandıkları varlığın yanına yerleştirir.

TABLE 4.3

CROW'S FOOT SYMBOLS

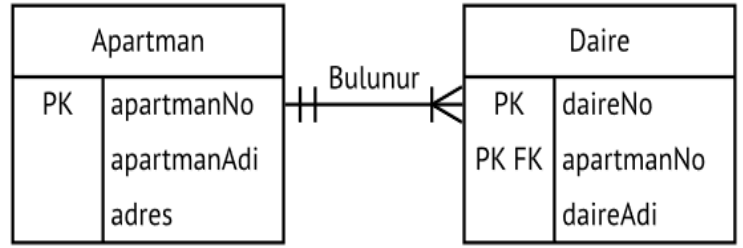
SYMBOL	CARDINALITY	COMMENT
O<	(0,N)	Zero or many; the "many" side is optional.
|<	(1,N)	One or many; the "many" side is mandatory.
||	(1,1)	One and only one; the "1" side is mandatory.
O|	(0,1)	Zero or one; the "1" side is optional.

Bağıntı (Relationship)
Var Olma Bağımlılığı(Existence Dependency)

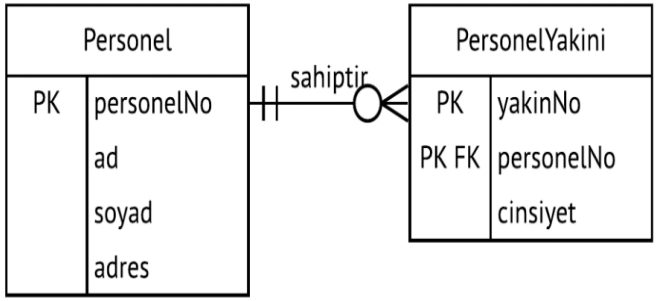
- Bir varlık veri tabanında sadece diğer bir bağıntılı varlıkla ilişkilendirildiğinde var olabiliyor ise **var olma-bağımlı (existence-dependent)** denir.
- Uygulama terimleri açısından, bir varlık zorunlu bir yabancı anahtara, yani boş olamayacak bir yabancı anahtar (foreign key) niteliğine sahipse var olma bağlıdır.
- Bir varlık, ilgili tüm varlıklarından ayrı olarak var olabiliyorsa, **var olma-bağımsızdır (existence-independent)** ve *güçlü varlık (strong entity) veya normal varlık (regular entity)* olarak adlandırılır.
- Chen gösterimi kavramsal modellemeye yöneliktir ve bu nedenle zayıf ve güçlü bağıntılar arasında ayrım yapmaz.
- Bağıntı gücü (relationship strength) kavramı, doğrudan Crow's Foot diyagramları için geçerlidir. Crow's Foot diyagramları ilişkisel veritabanları tasarlamak için yoğun şekilde kullanılır ve veri tabanı uygulamasını etkilediği için bağıntı gücünü anlamak önemlidir.

Bağıntı (Relationship)
Var Olma Bağımlılığı(Existence Dependency)

- Apartman-Daire
  - 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.



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



Bağıntı (Relationship)
Bağıntı Gücü (Relationship Strength)

Zayıf Bağıntı

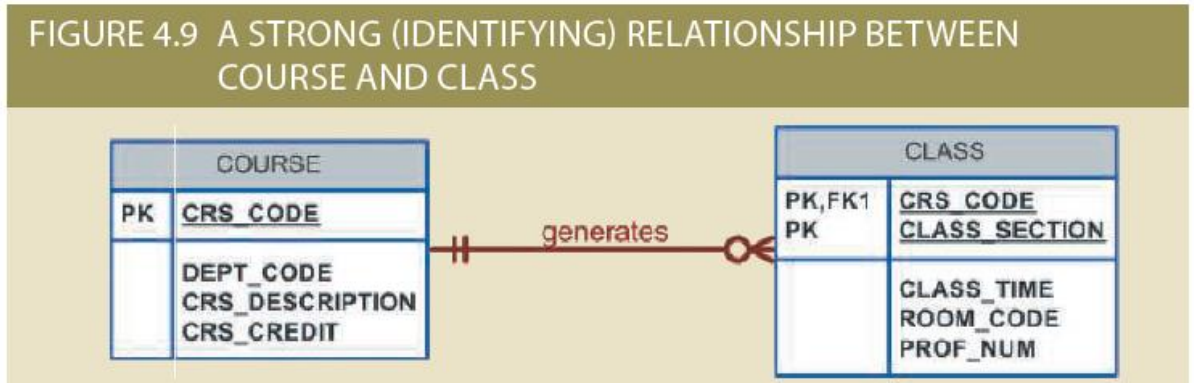
- Bağıntı kurulan varlığın birincil anahtarı içerisinde, bağıntı kuran varlığın birincil anahtar bilgisi yer almıyorsa "iki varlık arasında zayıf bağıntı vardır" denir.
- Crow's Foot gösterimi, varlıklar arasına kesikli bir ilişki çizgisi koyarak zayıf bir ilişkiyi gösterir.
- COURSE ve CLASS arasında zayıf bir ilişki vardır çünkü CRS_CODE (ana varlığın birincil anahtarı) CLASS varlığında yalnızca bir yabancı anahtardır.

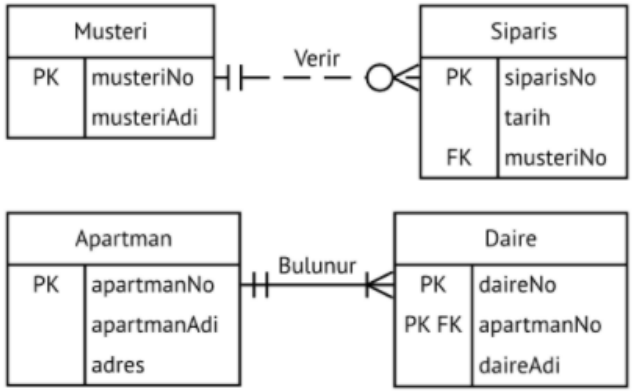
FIGURE 4.8 A WEAK (NON-IDENTIFYING) RELATIONSHIP BETWEEN COURSE AND CLASS

Bağıntı (Relationship)
Bağıntı Gücü (Relationship Strength)

Güçlü Bağıntı

- Bağıntı kurulan varlığın birincil anahtarı içerisinde, bağıntı kuran varlığın birincil anahtarı yer alıyorsa "iki varlık arasında güçlü bağıntı vardır" denir.
- CLASS varlığının birincil anahtarı (primary key) CRS_CODE ve CLASS_SECTION niteliklerinden oluşmaktadır.
- Crow's Foot notasyonu, varlıklar arasındaki düz bir çizgiyle güçlü ilişkiyi gösterir.





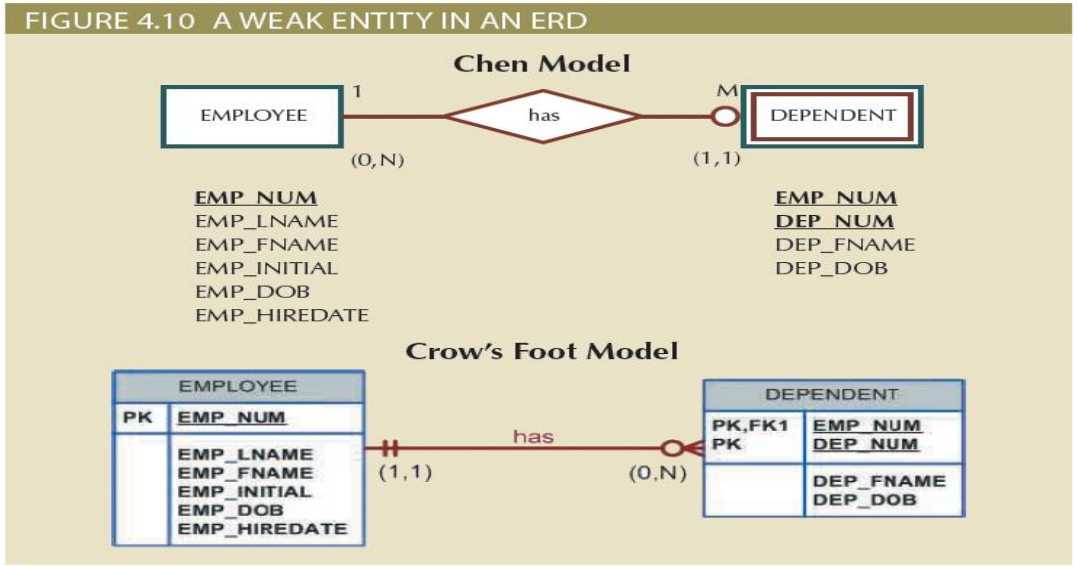
Zayıf Varlıklar (Weak Entity)

Aşağıdaki iki koşulu sağlayan varlıklar zayıf varlık olarak nitelendirilir.

1-Varlık, var olma-bağımlıdır (existence dependent).

2-Varlık, kısmen veya tamamen bağıntıdaki ana varlıktan türetilmiş bir birincil anahtara (primary key) sahiptir.

- Chen gösterimi, çift kenarlı bir varlık dikdörtgeni kullanarak zayıf varlığı tanımlar.
- Crow's Foot gösterimi, ilgili varlığın zayıf olup olmadığını belirtmek için bağıntı çizgisini ve PK / FK atamasını kullanır.
- Güçlü (tanımlayıcı) bir ilişki, ilgili varlığın zayıf olduğunu gösterir. Böyle bir ilişki, zayıf varlık tanımı için her iki koşulun da karşılandığı anlamına gelir

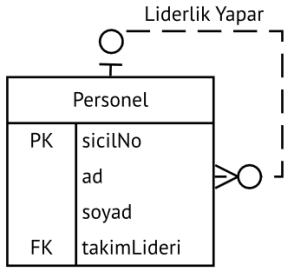


Bağıntı (Relationship)
Bağıntı Derecesi (Relationship Degree)

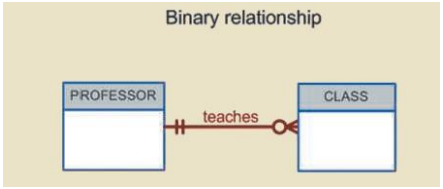
Bağıntı Derecesi; bir bağıntıyla ilişkili varlıkların veya katılımcıların sayısını gösterir.

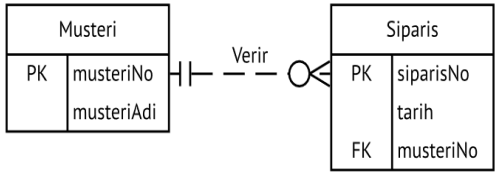
- Tekli (Unary) Bağıntı: Bir varlık kendisi ile bağıntılı (ilişkili) ise bu tür bir bağıntıya tekli bağıntı adı verilir.

  - Örneğin, Personel tablosu içerisindeki bir personel, sıfır veya daha fazla personelin aynı zamanda yöneticisidir. Bir personelin sıfır ya da bir yöneticisi olmalıdır.



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



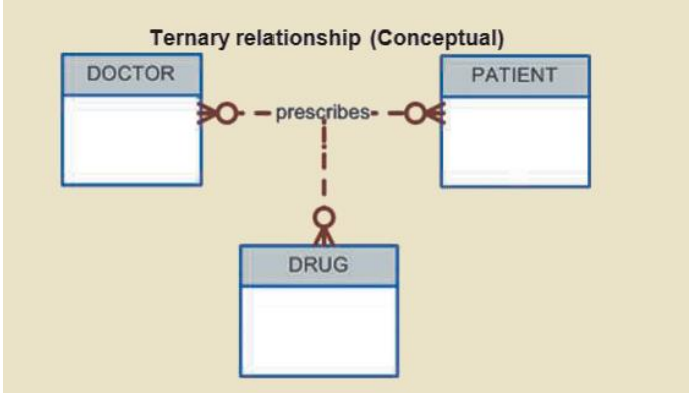


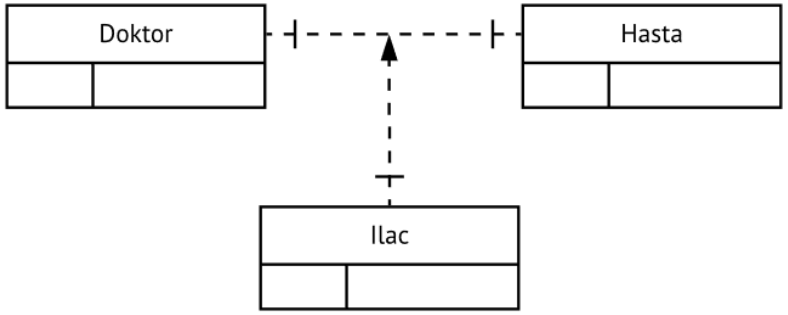
Bağıntı (Relationship)
Bağıntı Derecesi (Relationship Degree)

Bağıntı Derecesi; bir bağıntıyla ilişkili varlıkların veya katılımcıların sayısını gösterir.

- 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.

- Daha yüksek dereceler olmasına rağmen, nadirdir ve özel olarak adlandırılmamıştır.

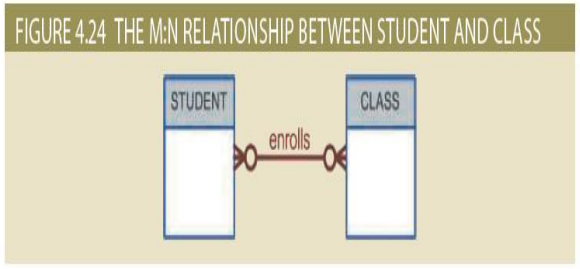


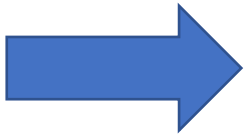


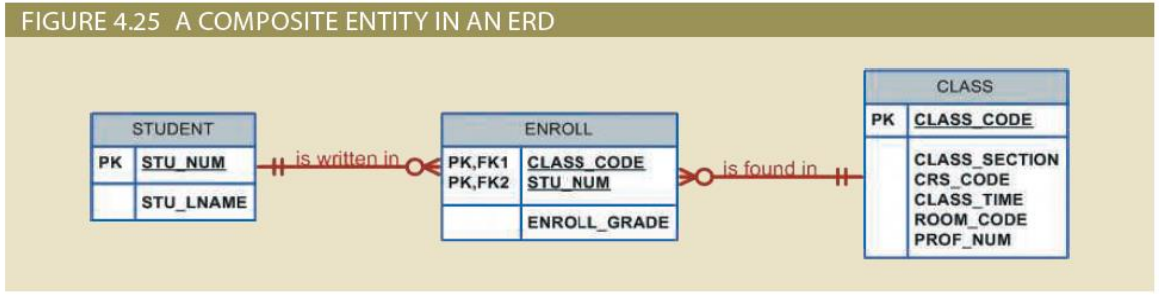
İlişkilendirilebilir - (Birleşik) – (Köprü) Varlık - (Associative - (Composite) – (Bridge) Entity)

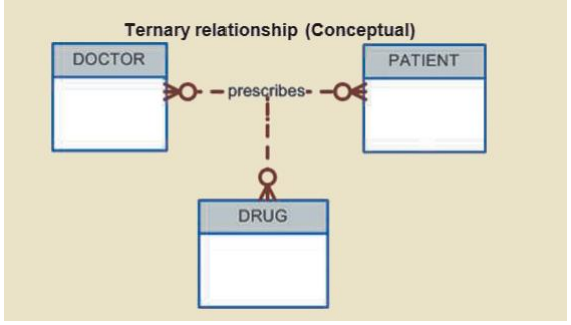
M: N ilişkisini özellikle ilişkisel modelde uygulamak, ek bir varlığın kullanılmasını gerektirir.

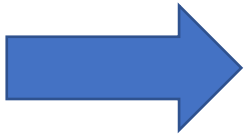
- VB modeli, iki veya daha fazla varlık arasındaki bir M: N ilişkisini temsil etmek için ilişkilendirilebilir varlığı kullanır.
- Bileşik veya köprü varlığı olarak da adlandırılan bu ilişkilendirilebilir varlık, ana varlıklarla 1: M ilişki içindedir ve her bir ana varlığın birincil anahtar özelliklerinden oluşur.
- İlişkilendirilebilir varlık kendi ek niteliklerine sahip olabilir.
- Crow's Foot notasyonu kullanılırken, köprü varlık, ebeveynler ile köprü varlık arasındaki düz(kesiksiz) bağıntı çizgileriyle gösterilen güçlü bir ilişki olarak tanımlanır.

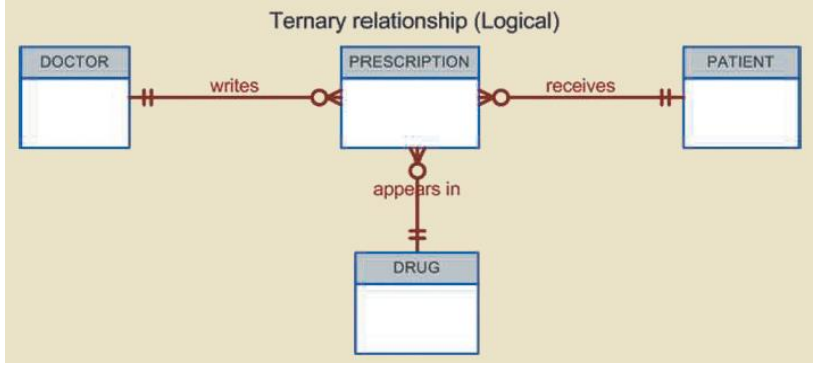












Varlık Bağıntı Diyagramı Geliştirmek

- 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 gelininceye kadar tekrarlanır.

Bölüm Özeti

- VBM, kavramsal veritabanını son kullanıcı tarafından görüntülendiği şekliyle temsil etmek için …… .. kullanır.
- VBM'nin ana bileşenleri ……., ……., ve …….
- VBD, Bağlılık (connectivity) ve kardinalite (cardinality) notasyonlarını içerir ve ayrıca ilişki gücünü, ilişki katılımını (isteğe bağlı veya zorunlu) ve ilişki derecesini (tekil, ikili veya üçlü gibi) gösterebilir.
- Bağlılık (Connectivity), …….. açıklar. Kardinalite (Cardinality), ilişkili varlığın oluşumuyla bağıntılı varlık oluşumu …… ifade eder.
- Bağlılık (Connectivity) ve Kardinalite (Cardinality) …… dayalı olarak belirlenir.
- VBM'de, kavramsal/mantıksal seviyede bir M:N ilişkisi geçerlidir. Ancak, ilişkisel bir veri tabanında VBM uygularken, M:N ilişkisi ……. kullanılarak bir …… ilişkisine dönüştürülmelidir.
- VBD'ler birçok farklı VBM'ne dayanabilir. Bununla birlikte, hangi model seçilirse seçilsin, modelleme mantığı aynı kalır. Hiçbir VBM, tüm gerçek dünya verilerini ve eylem kısıtlamalarını doğru bir şekilde gösteremeyeceğinden, iş kurallarının en azından bazılarının uygulanmasını artırmak için uygulama yazılımı kullanılmalıdır.
- Birleştirilmiş Modelleme Dili (UML) sınıf diyagramları, bir veri modelindeki statik veri yapılarını temsil etmek için kullanılır. UML sınıfında ve ER diyagramlarında kullanılan semboller çok benzerdir. UML sınıf diyagramları …… soyutlama düzeyinde veri modellerini göstermek için kullanılabilir.
- Veri tabanı tasarımcılarının ayrıntılı ve derinlemesine veri modelleme kuralları bilgisine sahip olması gerekir. Tasarım sürecini baştan sona belgelemek de önemlidir, bu da tasarım sürecini yolunda tutmaya yardımcı olur ve gelecekte kolay değişikliklere izin verir.

Referenslar

- Carlos Coronel, Steven Morris, and Peter Rob, Database Systems: Design, Implementation, and Management, Cengage Learning.
- https://github.com/celalceken/DatabaseManagementSystems