

# **Gelişmiş Veri Modelleme** **(Extended Entity** **Relationship)**

# Öğrenme Hedefleri

**Bu bölümü tamamladıktan sonra şunları yapabileceksiniz:**

- Ana genişletilmiş varlık ilişkisi (EER) modelini açıklama
- Varlık kümeleri birden çok varlığı ve ilişkiyi temsil etmek için nasıl kullanılır?
- İyi birincil anahtarların özellikleri ve bunların nasıl seçileceği
- Özel veri modelleme vakaları için esnek çözümler nasıl kullanılır?

# Genişletilmiş Varlık İlişki Modeli (EERM)

- Orijinal varlık ilişkisi (ER) modeline daha fazla anlamsal yapı eklemek gerekebilir.
- EER diyagramı (EERD): EER modelini kullanır.

# Varlık Üst Tipleri (supertype) ve Alt Türleri (subtype)

- ❖ **Varlık üst türü:** Bir veya daha fazla varlık alt türüyle ilişkili genel varlık türü
  - Örneğin; Çalışan ortak özellikler içerir
- ❖ **Varlık alt türü:** Her varlık alt türünün benzersiz özelliklerini içerir.
  - Örneğin; Pilot, Mekanik ve Muhasebeci
- ❖ **Kullanımı kriterleri belirleme**
  - *Kullanıcının ortamında varlığın farklı, tanımlanabilir türleri olmalıdır.*
  - **Farklı türdeki örneklerin her biri, o tür örneğe özgü bir veya daha fazla özneliğe sahip olmalıdır.**

- ❑ Varlıklar gruplandırılmazsa, aşağıdaki durumlar (sorunlar) oluşur:
- ❑ Veritabanındaki özniteliklerde **gereksiz** veya **null** değerler oluşur.
  - ❑ Örneğin; Çalışan varlığının veritabanında aşağıdaki gibi boş (null) kayıtlar oluşur.
- ❑ Belirli bir çalışan türü, o çalışan türüne özgü ilişkilere katılamaz.
  - ❑ Örneğin; pilota özel iş atamaları yapılmayacaktır.

FIGURE 5.1 NULLS CREATED BY UNIQUE ATTRIBUTES

Database name: Ch05\_AirCo

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_LICENSE	EMP_RATINGS	EMP_MED_TYPE	EMP_HIRE_DATE
100	Kolmycz	Xavier	T				15-Mar-88
101	Lewis	Marcos		ATP	SEL/MEL/Instr/CFII	1	25-Apr-89
102	Vandam	Jean					20-Dec-93
103	Jones	Victoria	R				28-Aug-03
104	Lange	Edith		ATP	SEL/MEL/Instr	1	20-Oct-97
105	Williams	Gabriel	U	COM	SEL/MEL/Instr/CFI	2	08-Nov-97
106	Duzak	Mario		COM	SEL/MEL/Instr	2	05-Jan-04
107	Diante	Venite	L				02-Jul-97
108	Wiesenbach	Joni					18-Nov-95
109	Travis	Brett	T	COM	SEL/MEL/SES/Instr/CFII	1	14-Apr-01
110	Genkazi	Stan					01-Dec-03

### Employee Entity :

Kayıtlar: Pilot, Mekanik  
ve Muhasebeci

\* Varlıkları gruplandırma (Uzmanlık Hiyerarşisi), olumsuz durumlardan (boş kayıtlar vb.) kaçınmaya izin verir.

# Özelleştirme Hiyerarşisi

Üst düzey ve alt düzey varlık alt türlerinin düzenini gösterir.

İlişkiler "is-a" ilişkileri olarak tanımlanır.

Alt tür, bir üst tür bağlamında var olur.

Her alt tipin doğrudan ilişkili olduğu bir üst türü vardır.

Süper tipin birçok alt türü olabilir.

# Özelleştirme Hiyerarşisi

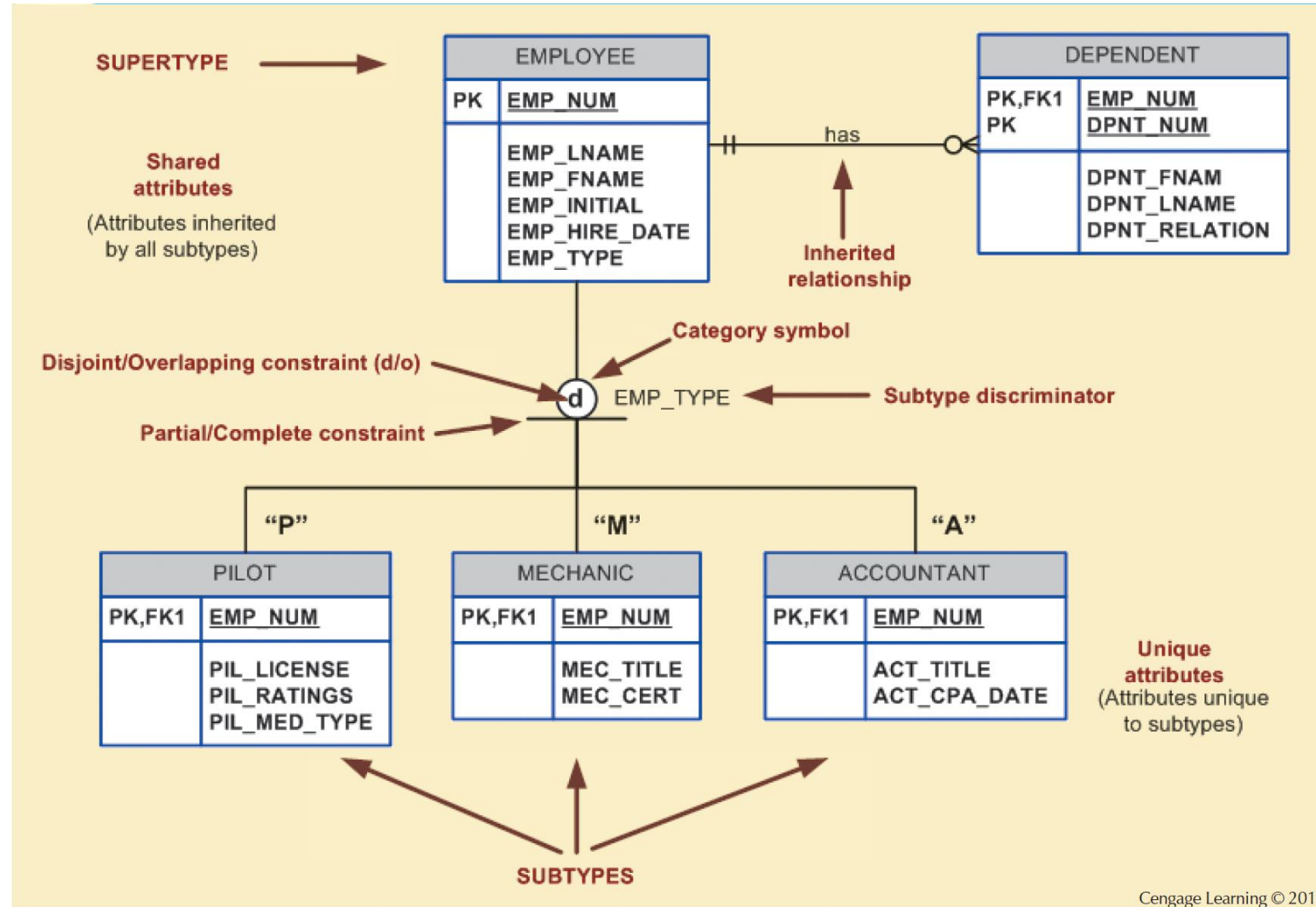
**Aşağıdaki katkıları sağlar.**

Destek özniteliği devralma

Alt tür ayırıcı olarak bilinen özel bir süper tür özniteliği tanımlama.

Ayrık (ayrık)/çakışan (kesişim) kısıtlamaları ve Tamlık (toplam veya kısmi) Kısıtlamaları tanımlama.

# Özelleştirme Hiyerarşisi





# Miras (Inheritance)



**Bir varlık alt türünün, üst türün özniteliklerini ve ilişkilerini devralmasını sağlar.**

**Tüm varlık alt türleri, birincil anahtar özniteliklerini üst türlerinden devralır.**

**Uygulama düzeyinde, üst tür ve alt türü (türleri) 1: 1 ilişkisini korur.**

**Varlık alt türleri, üst tür varlığının katıldığı tüm ilişkileri devralır.**

**FIGURE  
5.3**

## The EMPLOYEE-PILOT supertype-subtype relationship

**Table Name: EMPLOYEE**

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIRE_DATE	EMP_TYPE
100	Kolmycz	Xavier	T	15-Mar-88	
101	Lewis	Marcos		25-Apr-89	P
102	Vandam	Jean		20-Dec-93	A
103	Jones	Victoria	R	28-Aug-03	
104	Lange	Edith		20-Oct-97	P
105	Williams	Gabriel	U	08-Nov-97	P
106	Duzak	Mario		05-Jan-04	P
107	Diante	Verite	L	02-Jul-97	M
108	Wiesenbach	Joni		18-Nov-95	M
109	Travis	Brett	T	14-Apr-01	P
110	Genkazi	Stan		01-Dec-03	A

**Table Name: PILOT**

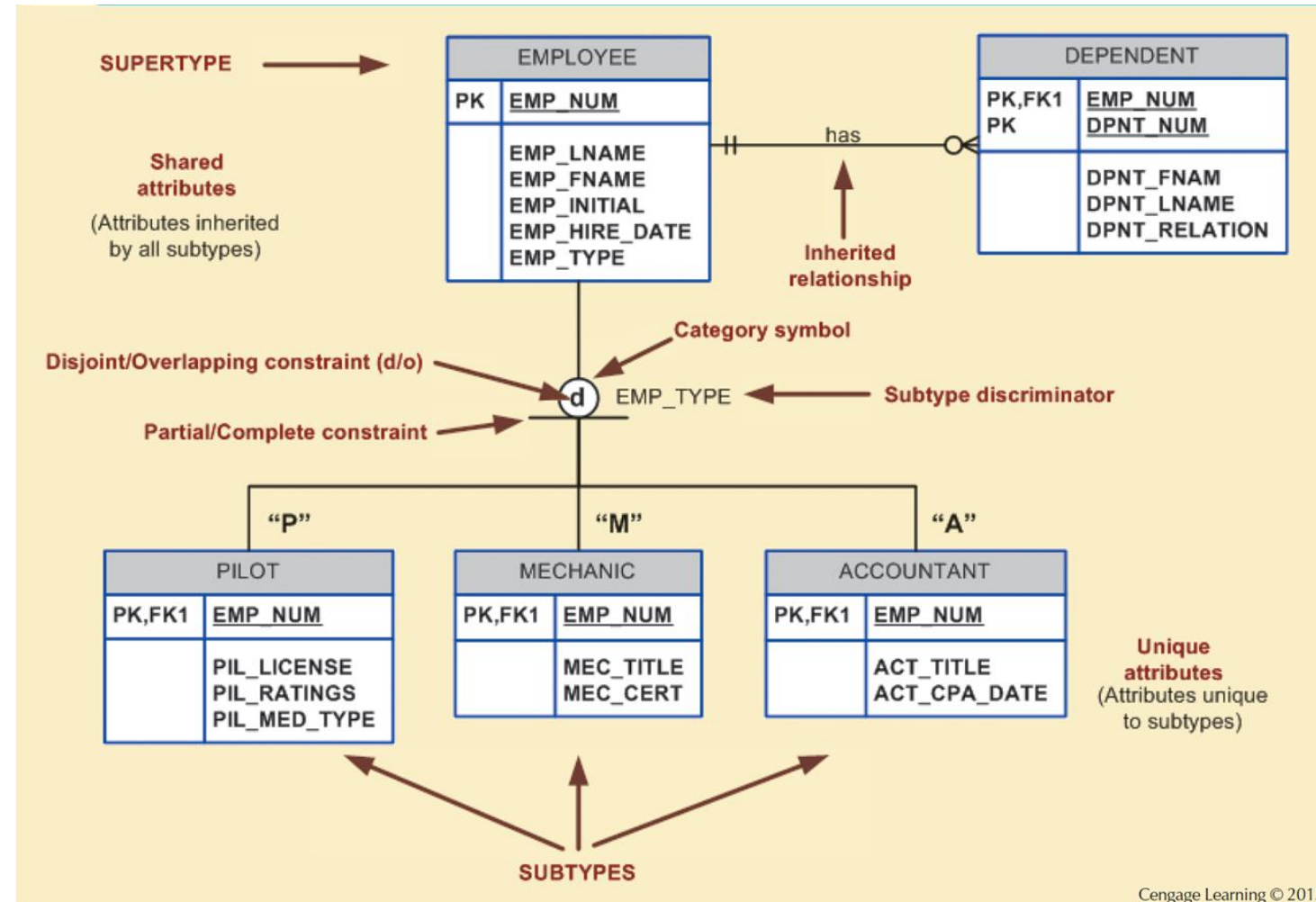
EMP_NUM	PIL_LICENSE	PIL_RATINGS	PIL_MED_TYPE
101	ATP	SEL/MEL/Instr/CFI	1
104	ATP	SEL/MEL/Instr	1
105	COM	SEL/MEL/Instr/CFI	2
106	COM	SEL/MEL/Instr	2
109	COM	SEL/MEL/SES/Instr/CFI	1

## Alt Tip Ayırıcı(Subtype Discriminator )

- ❑ Üst tür oluşumunun hangi **varlık alt türüyle ilişkili olduğunu belirler.**
- ❑ Varsayılan karşılaştırma **koşulu, eşitlik karşılaştırmasıdır.**

Yandaki şekilde

- EMP\_TYPE "P" değerine sahipse PİLOT alt türüyle ilişkilidir.
- EMP\_TYPE değeri "M" ise, üst tür bir MECHANIC alt türüyle ilişkilidir.
- EMP\_TYPE değeri "A" ise, üst tür MUHASEBECİ alt türüyle ilişkilidir.



# Ayrık ve Çakışan Kısıtlamalar

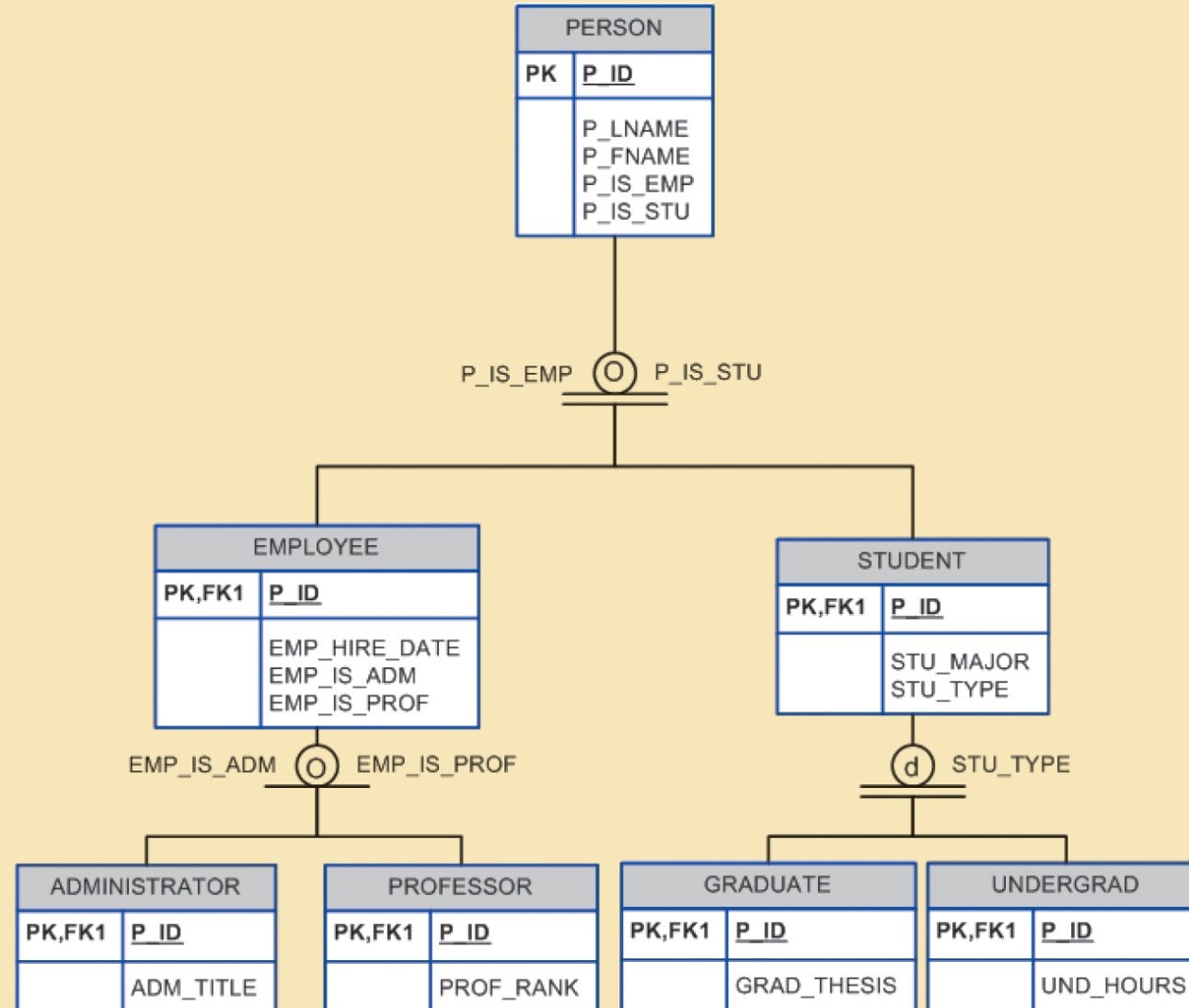
- **Ayrık alt türler:** Üst tür varlık kümesinin benzersiz bir alt kümesini içerir.
- Örtüşmeyen alt türler olarak bilinir
- Uygulamada, üst türdeki alt tür ayırıcı özniteliğinin değerini temel alır.

Örneğin; The STUDENT, MEZUN (GRADUATE) veya UNDERGRAD (LİSANS) alt türünü içerir.

- **Çakışan alt türler:** Üst tür varlık kümesinin benzersiz olmayan alt kümelerini içerir.
  - Uygulamada, her alt tür için bir ayırıcı özniteliğin kullanılmasını gerektirir.

Örneğin; **EMPLOYEE** üst varlıkta **PERSON**, **PROFESSOR** ve **ADMINISTRATOR** çakışabilir.

# Çakışan Alt Türlere Sahip Özelleştirmeye Hiyerarşisi



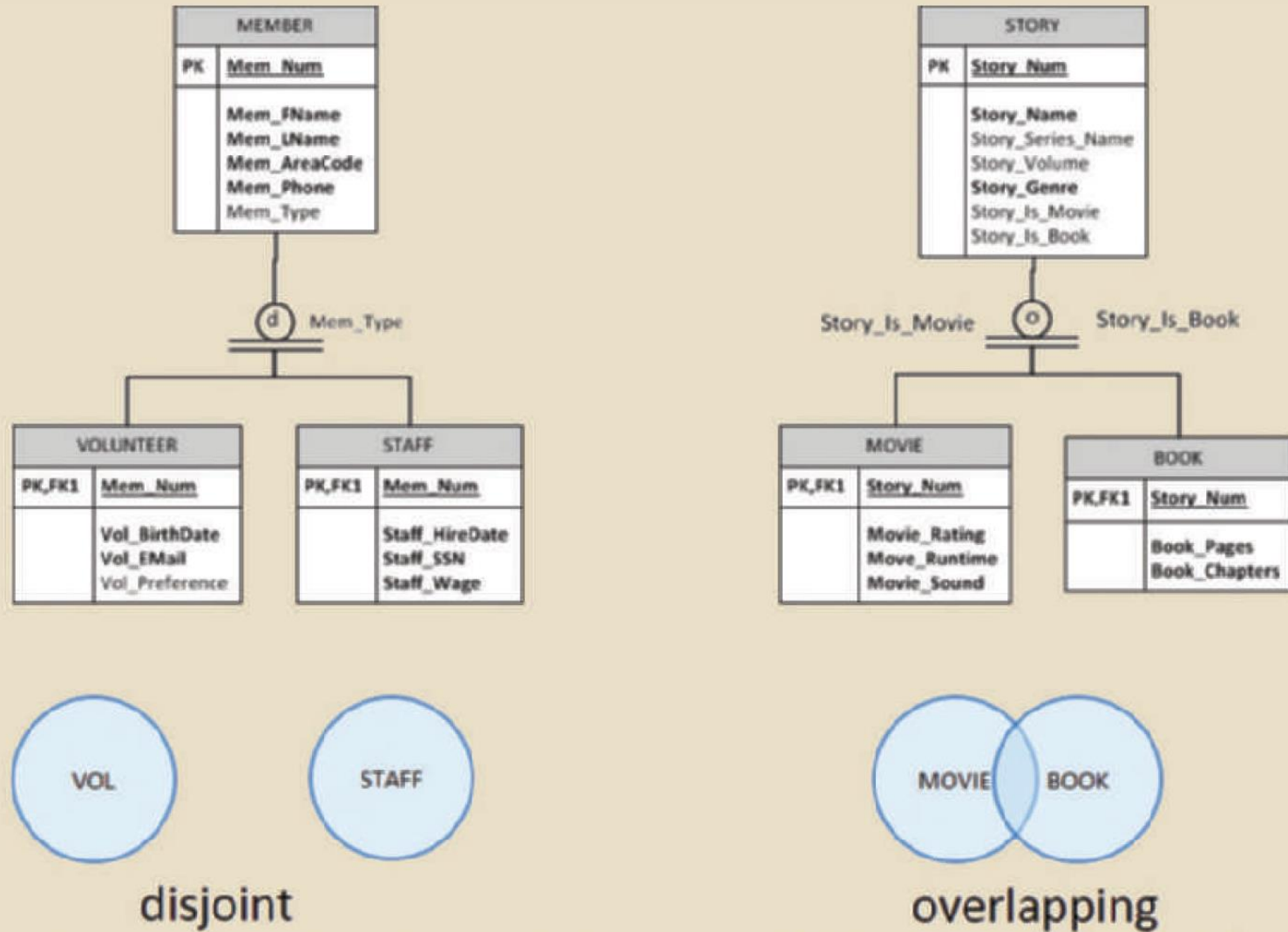
# Çakışan Alt Türlerle Sahip Ayırıcı Öznitelikler

TABLE  
5.1

Discriminator Attributes with Overlapping Subtypes

DISCRIMINATOR ATTRIBUTES		COMMENT
Professor	Administrator	
"Y"	"N"	The Employee is a member of the Professor subtype.
"N"	"Y"	The Employee is a member of the Administrator subtype.
"Y"	"Y"	<u>The Employee is both a Professor and an Administrator.</u>

FIGURE 5.5 DISJOINT AND OVERLAPPING SUBTYPES





# Tamlık Kısıtlaması

- Her üst tür oluşumunun aynı zamanda en az bir alt türün üyesi olması gerekip gerekmediğini belirtir.

- **Türler**



- **Kısmi bütünlük (kategorizasyon):** Her üst tür oluşumu bir alt türün üyesi değildir.

Örneğin; Müşteri, *Normal* veya *Özel* alt türü olarak tanımlanır. Ancak *Diğer* alt türü tasarımda gösterilmez.

- **Toplam bütünlük:** Her süper tip oluşumu herhangi birinin üyesi olmalıdır.

Örneğin; **Müşterinin** cinsiyeti, **Erkek** veya **Kadın** alt tipidir.

# Uzmanlık Hiyerarşisi Kısıtlama Senaryoları

UZMANLAŞTIRMA HİYERARŞİSİ KISITLAMA SENARYOLARI		
TÜR	AYRIK KISITLAMA	ÖRTÜŞEN KISITLAMA
	<p>Süper türün isteğe bağlı alt türleri vardır.</p> <p>Alt tür ayırıcısı boş olabilir.</p> <p>Alt tür kümeleri benzersizdir.</p>	<p>Süper türün isteğe bağlı alt türleri vardır.</p> <p>Alt tür ayırıcılar boş olabilir.</p> <p>Alt tür kümeleri benzersiz değildir.</p>
	<p>Her üst tip oluşumu sadece bir alt tipin üyesidir.</p> <p>Alt tür ayırıcısı boş olamaz.</p> <p>Alt tür kümeleri benzersizdir.</p>	<p>Her üst tip oluşumu, en az bir alt tipin üyesidir.</p> <p>Alt tür ayırıcılar boş olamaz.</p> <p>Alt tür kümeleri benzersiz değildir.</p>

# Özelleştirme ve Genelleme

## Özelleştirme

- **Top-down süreç**
- Daha üst düzey *bir varlık üst türünden* daha düşük düzeyli, daha spesifik varlık alt türlerini tanımlar.
- Alt türlerin benzersiz özelliklerini ve ilişkilerini gruplandırmaya dayalıdır.

## Genelleştirme

- **Bottom-up süreç**
- Alt düzey varlık alt türlerinden daha yüksek düzeyli, daha genel bir varlık üst türü tanımlar.
- Alt türlerin ortak özelliklerini ve ilişkilerini gruplandırmaya dayalıdır.

# Özelleştirme ve Genelleme

## ■ Özelleştirme:

- Örneğin, orijinal çalışan üst türünden birden fazla varlığı tanımlamak için özelleştirilmesi için *pilot, muhasebeci vb.* tablolar oluşturuldu.

## ■ Genelleştirme

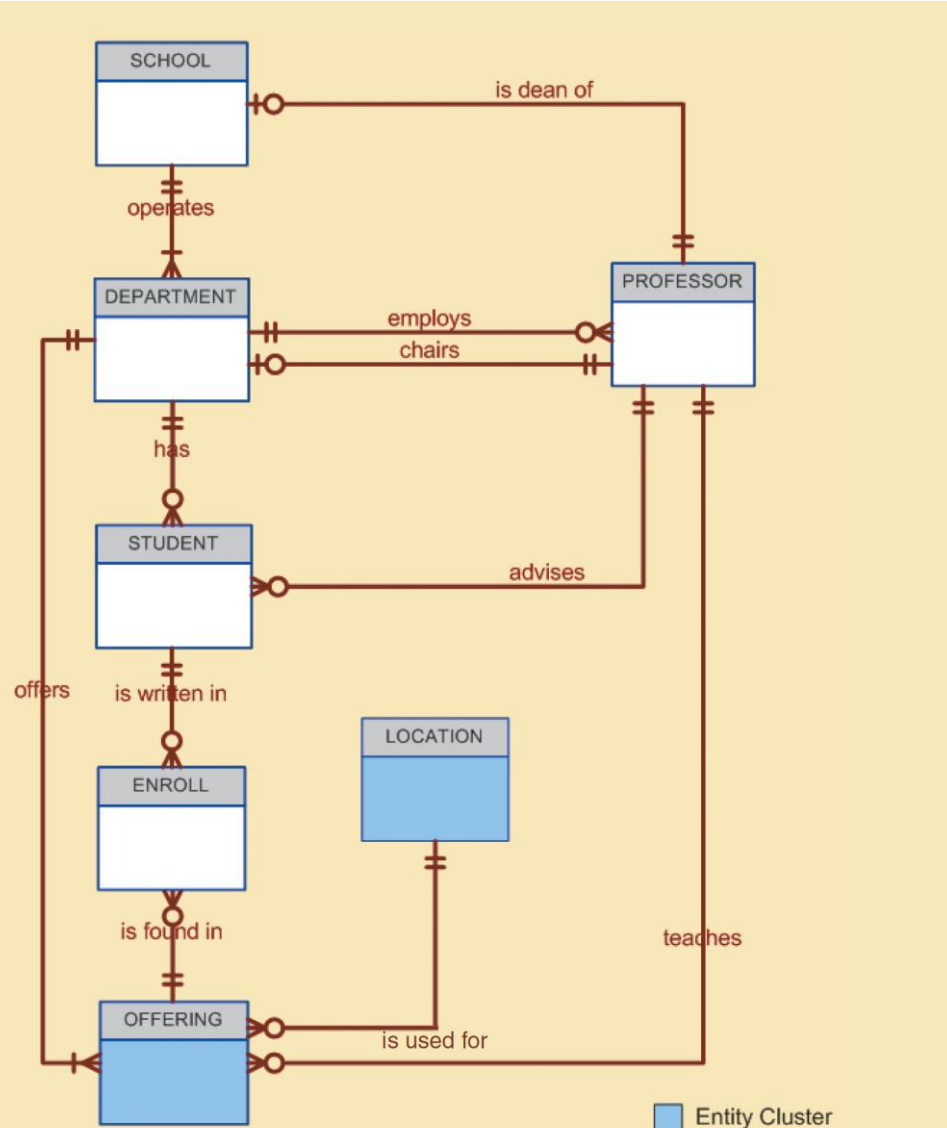
- Örneğin, birden fazla müzik aleti türü tanımlayabilirsiniz: piyano, keman ve gitar.
- Genelleme yaklaşımını kullanarak, birden çok alt türün ortak özelliklerini tutmak için bir "dize aracı" varlık üst türü tanımlayabilirsiniz.

## Varlık Kümesi

- Varlık kümesi, ERD'deki **birden çok varlığı ve ilişkiyi temsil etmek için kullanılan "sanal" bir varlık türüdür.**
- **Bir varlık kümesi, birbiriyle ilişkili birden çok varlığın tek bir soyut varlık nesnesinde birleştirilmesiyle oluşturulur.**

- **Bir varlık kümesi, aslında ERD'de bir varlık olmadığı anlamında 'sanal' veya 'soyut' olarak kabul edilir.**
- **Bunun yerine, ERD'yi basitleştirmek ve böylece okunabilirliğini artırmak amacıyla **birden fazla varlığı ve ilişkiyi temsil etmek için kullanılan geçici bir varlıktır.****
  - **SUNUM; DÖNEM, KURS ve SINIF varlıklarını ve ilişkilerini gruplandırılması**
  - **LOKASYON; ODA ve BİNA varlıklarını ve ilişkilerini gruplandırılması için oluşturulmuştur.**

# Varlık Kümelerini Kullanan Tiny College ERD



Yandaki şekilde varlıkların öz nitelikleri gösterilmemiştir.

Varlık kümeleri kullanılırken, birleştirilmiş varlıkların temel öz nitelikleri artık kullanılamaz.

Anahtar öz nitelikleri olmadan, birincil anahtar devralma kuralları değişir.

# Doğal Anahtarlar veya Doğal Tanımlayıcı

- ❑ Gerçek dünyadaki nesneleri benzersiz şekilde tanımlamak için doğal tanımlayıcı kullanılır.
  - Son kullanıcılara tanıdık gelen ve günlük hayatta işte kullanılan kelimeler kullanılır.
- ❑ Doğal tanımlayıcı olarak da bilinir.
  - Modellenen varlığın birincil anahtarı olarak kullanılır.



# Birincil Anahtarlar

- Tek öznitelik veya her varlık örneğini benzersiz şekilde tanımlayan özniteliklerin bir kombinasyonudur.
- Varlık bütünlüğünü garanti eder.
- İlişkileri uygulamak için yabancı anahtarlarla çalışır.

# Birincil Anahtarlar (2)

TABLE 5.3

## DESIRABLE PRIMARY KEY CHARACTERISTICS

PK CHARACTERISTIC	RATIONALE
Unique values	The PK must uniquely identify each entity instance. A primary key must be able to guarantee unique values. It cannot contain nulls.
Nonintelligent	The PK should not have embedded semantic meaning other than to uniquely identify each entity instance. An attribute with embedded semantic meaning is probably better used as a descriptive characteristic of the entity than as an identifier. For example, a student ID of 650973 would be preferred over Smith, Martha L. as a primary key identifier.
No change over time	If an attribute has semantic meaning, it might be subject to updates, which is why names do not make good primary keys. If Vickie Smith is the primary key, what happens if she changes her name when she gets married? If a primary key is subject to change, the foreign key values must be updated, thus adding to the database work load. Furthermore, changing a primary key value means that you are basically changing the identity of an entity. In short, the PK should be permanent and unchangeable.
Preferably single-attribute	A primary key should have the minimum number of attributes possible (irreducible). Single-attribute primary keys are desirable but not required. Single-attribute primary keys simplify the implementation of foreign keys. Having multiple-attribute primary keys can cause primary keys of related entities to grow through the possible addition of many attributes, thus adding to the database workload and making (application) coding more cumbersome.
Preferably numeric	Unique values can be better managed when they are numeric, because the database can use internal routines to implement a counter-style attribute that automatically increments values with the addition of each new row. In fact, most database systems include the ability to use special constructs, such as Autonumber in Microsoft Access, sequence in Oracle, or uniqueidentifier in MS SQL Server to support self-incrementing primary key attributes.
Security-compliant	The selected primary key must not be composed of any attribute(s) that might be considered a security risk or violation. For example, using a Social Security number as a PK in an EMPLOYEE table is not a good idea.

# Birincil Anahtarlar (2)

## İSTENİLEN BİRİNCİL ANAHTAR ÖZELLİKLER

PK KARAKTERİSTİK	GEREKÇE
Benzersiz değerler	PK, her varlık örneğini benzersiz şekilde tanımlamalıdır. Birincil anahtar, benzersiz değerleri garanti edebilmelidir. Boş değerler içeremez.
Anlamsal olmayan	PK, her bir varlık örneğini benzersiz şekilde tanımlamak dışında gömülü anlamsal anlam içermemelidir. Gömülü semantik anlamı olan bir nitelik, muhtemelen bir tanımlayıcıdan ziyade varlığın tanımlayıcı bir özelliği olarak kullanılır. Örneğin, bir 650973 öğrenci kimliği, birincil anahtar tanımlayıcısı olarak Smith, Martha L.'ye tercih edilir.
Zamanla değişiklik yok	Bir özniteliğin <b>anlamsal anlamı varsa, güncellemelere tabi olabilir</b> , bu nedenle isimler iyi birincil anahtarlar oluşturmaz. <i>Vickie Smith</i> birincil anahtarsa, <b>evlendiğinde adını değiştirirse ne olur?</b> Bir birincil anahtar değişebilirse, yabancı anahtar değerlerinin güncellenmesi gerekir, böylece veritabanı iş yüküne eklenir. <b>Ayrıca, birincil anahtar değerini değiştirmek, temelde bir varlığın kimliğini değiştirdiğiniz anlamına gelir.</b> Kısacası, PK kalıcı ve değiştirilemez olmalıdır.

# Birincil Anahtarlar (2)

## İSTENİLEN BİRİNCİL ANAHTAR ÖZELLİKLER

PK KARAKTERİSTİK	GEREKÇE
Tercihen tek nitelik	Birincil anahtar, mümkün olan en az sayıda özniteliğe sahip olmalıdır (indirgenemez). Tek öznitelikli birincil anahtarlar istenir ancak gerekli değildir. <b>Birden çok özniteliğe sahip birincil anahtarlara sahip olmak, birçok öznenin olası eklenmesi yoluyla ilgili varlıkların birincil anahtarlarının büyümesine neden olabilir, böylece veritabanı iş yüküne eklenir ve (uygulama) kodlaması daha hantal hale gelir.</b>
Tercihen sayısal	Benzersiz değerler sayısal olduklarında daha iyi yönetilebilir, çünkü veritabanı, her yeni satırın eklenmesiyle değerleri otomatik olarak artıran bir sayaç stili özniteliği uygulamak için dahili rutinleri kullanabilir.
Güvenlik uyumlu	Seçilen birincil anahtar, güvenlik riski veya ihlali olarak değerlendirilebilecek herhangi bir özellikten oluşmamalıdır. Örneğin, bir EMPLOY EE tablosunda bir <b>Sosyal Güvenlik numarasını PK olarak kullanmak iyi bir fikir değildir.</b>

# Bileşik Birincil Anahtarların Kullanımı

**Bileşik birincil anahtarlar özellikle iki durumda kullanışlıdır:**

**1. Her birincil anahtar kombinasyonuna M:N ilişkisinde bir kez izin verilir.**

Örneğin; ÖĞRENCİ varlık kümesi ve CLASS varlık kümesi

**2. Zayıf varlıkların tanımlayıcıları olması durumu**

**Zayıf varlığın ana varlıkla güçlü bir tanımlayıcı ilişkisi vardır.**

Örneğin; MÜŞTERİ varlık kümesi ve bir FATURA varlık kümesi

Database name: Ch05\_Tinycollege

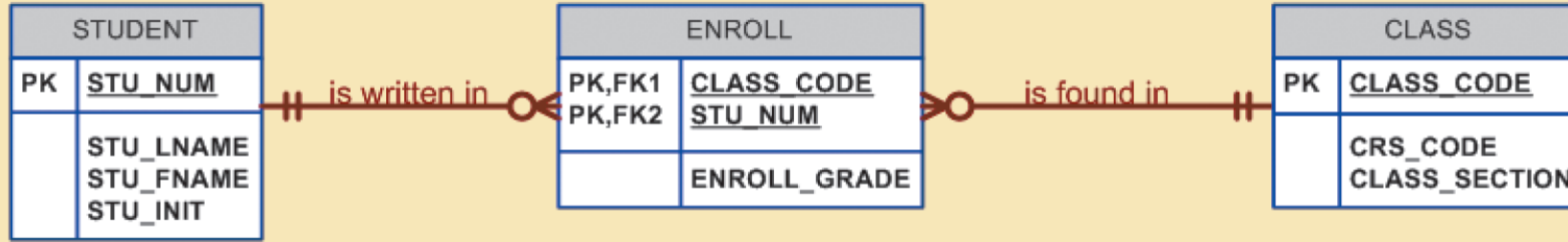


Table name: STUDENT  
(first four fields)

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT
321452	Bowser	William	C
324257	Smithson	Anne	K
324258	Brewer	Juliette	
324269	Oblonski	Walter	H
324273	Smith	John	D
324274	Katinga	Raphael	P
324291	Robertson	Gerald	T
324299	Smith	John	B

Table name: ENROLL

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS  
(first three fields)

CLASS_CODE	CRS_CODE	CLASS_SECTION
10012	ACCT-211	1
10013	ACCT-211	2
10014	ACCT-211	3
10015	ACCT-212	1
10016	ACCT-212	2
10017	CIS-220	1
10018	CIS-220	2
10019	CIS-220	3
10020	CIS-420	1
10021	QM-261	1
10022	QM-261	2
10023	QM-362	1
10024	QM-362	2
10025	MATH-243	1

Cengage Learning © 2015

- Yukarıdaki şekilde gösterildiği gibi, bileşik birincil anahtar otomatik olarak yinelenen değerler olmamasını sağlama avantajını sağlar.
- Bir başka ifadeyle aynı öğrencinin aynı sınıfa birden fazla kez kaydolamayacağını garanti eder.

Bir ana varlıkla güçlü bir tanımlama ilişkisindeki zayıf varlık normalde iki durumdan birini temsil etmek için kullanılır:

**1. Bir varlığın varoluşu diğer varlığın oluşturulması ile temsil edilir.**

**2. Veri modelinde güçlü bir tanımlayıcı ilişki içinde iki ayrı varlık olarak temsil edilir.**

Örneğin, gerçek dünyadaki fatura nesnesi bir veri modelinde iki varlıkla temsil edilir:

**INVOICE ve LINE.**

Açıkçası, **LINE** varlığı gerçek dünyada bağımsız bir nesne olarak değil, bir INVOICE'nin parçası olarak var olur.

# Vekil/Yedek (Surrogate) Anahtarlar

Vekil anahtar, varlık örneklerinin tanımlanmasını basitleştirmek için veritabanı tasarımcısı tarafından oluşturulan birincil anahtardır.

- ❑ Vekil anahtar, aşağıdaki durumlarda kullanılma ihtiyacı doğar:
  - Doğal anahtar olaması
  - Seçilen aday anahtar, anlamsal içeriğe sahip veya çok uzun olması
- ❑ Söz konusu varlığın aday anahtarının düzgün performans gösterdiğinden emin olmayı gerektirir.
  - Benzersiz ve boş olmama kısıtı konulur.



- Örneğin, küçük partiler için oda kiralayan bir park rekreasyon tesisi (etkinlik, etkinlik)

■

## DATA USED TO KEEP TRACK OF EVENTS

DATE	TIME_START	TIME_END	ROOM	EVENT_NAME	PARTY_OF
6/17/2018	11:00 a.m.	2:00 p.m.	Allure	Burton Wedding	60
6/17/2018	11:00 a.m.	2:00 p.m.	Bonanza	Adams Office	12
6/17/2018	3:00 p.m.	5:30 p.m.	Allure	Smith Family	15
6/17/2018	3:30 p.m.	5:30 p.m.	Bonanza	Adams Office	12
6/18/2018	1:00 p.m.	3:00 p.m.	Bonanza	Boy Scouts	33
6/18/2018	11:00 a.m.	2:00 p.m.	Allure	March of Dimes	25
6/18/2018	11:00 a.m.	12:30 p.m.	Bonanza	Smith Family	12

**EVENT** (DATE, TIME\_START, TIME\_END, ROOM, EVENT\_NAME, PARTY\_OF)

Aşağıdaki seçeneklerden birini önerebilirsiniz: (DATE, TIME\_START, ROOM) or (DATE, TIME\_END, ROOM)

Bileşik birincil anahtarı (DATE, TIME\_START, ROOM) olabilir.

Ardından, bir ETKİNLİK birçok KAYNAK (tablolar, projektörler, PC'ler ve standlar gibi) kullanabileceğini ve aynı KAYNAK'ın birçok ETKİNLİK için kullanılabileceğini belirlersiniz.

❑ **RESOURCE** (RSC\_ID, RSC\_DESCRIPTION, RSC\_TYPE, RSC\_QTY, RSC\_PRICE)

İş kuralları göz önüne alındığında:

RESOURCE ve EVENT arasındaki M:N ilişkisi, EVNTRSC bileşik varlığı aracılığıyla aşağıdaki gibi bileşik birincil anahtarla temsil edilir:

❑ **EVNTRSC** (DATE, TIME\_START, ROOM, RSC\_ID, QTY\_USED)

Artık uzun, dört öznitelikli bileşik birincil anahtarınız var.

**Vekil anahtar özellikle doğal anahtar olmadığında,  
seçilen aday anahtar anlamsal içeriğe sahip olduğunda  
veya seçilen aday anahtar çok uzun veya hantal  
oldüğunda yararlıdır.**

**Vekil anahtar **EVENT** varlığında atanır:**

- **Oluşturulan birincil anahtar,**
- **Genellikle sayısal**
- **Otomatik artımlı**

# Tasarım Örnekleri: Esnek Öğrenme Veritabanı Tasarımı

Bu bölümde **esnek tasarımların önemini**, birincil anahtarların doğru tanımlanmasını ve yabancı anahtarların yerleştirilmesini vurgulayan dört özel tasarım durumu sunulmaktadır.

# Tasarım Örnekleri: Esnek Öğrenme Veritabanı Tasarımı

## 1:1 İlişkileri Uygulama

- ❑ Zaman Değişkeni Verilerinin Geçmişini Korumak
- ❑ Tuzaklar
- ❑ Gereksiz İlişkiler

# Tasarım Örneği 1: 1:1 İlişkileri Uygulama

- **Yabancı anahtarlar**, ilişkileri ilişkisel modelde düzgün bir şekilde uygulamak için birincil anahtarlarla birlikte çalışır.
- **Kural**
  - *Ana varlığın birincil anahtarını bağımlı varlığa yabancı anahtar olarak yerleştirilir.*
- **Yabancı anahtarı seçme ve yerleştirme seçenekleri:**
  - **Her iki varlığa da yabancı anahtar yerleştirilmesi: Bu çözüm önerilmez.**
  - **Varlıklardan birine yabancı bir anahtar yerleştirilir: Tercih edilen çözüm budur.**

# 1:1 İlişkisinde Yabancı Anahtar Seçimi

Tercih edilen çözüm budur, ancak bir soru kalır:

Yabancı anahtar olarak hangi birincil anahtar kullanılmalıdır?

Bu soru hakkında üç vaka. Aşağıda bu durumlar gösterilmiştir.

## 1:1 İLİŞKİDE YABANCI ANAHTAR SEÇİMİ

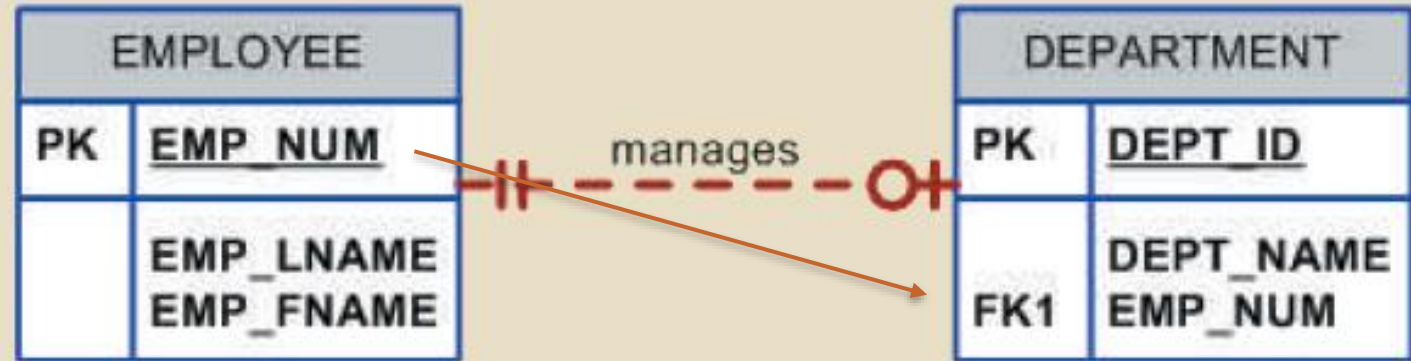
DURUM	ER İLİŞKİSİ KISITLAMALARI	AKSİYON
I	Bir taraf zorunlu, diğer taraf opsiyonel	Zorunlu taraftaki PK'yı isteğe bağlı taraftaki varlıkta <b>FK olarak yerleştirin</b> ve <b>FK'yi zorunlu hale getirin</b> .
II	Her iki taraf da isteğe bağlıdır.	<b>En az boş değere neden olan FK'yi seçin</b> veya <b><u>FK'yi (ilişki) rolünün oynandığı varlığa yerleştirin.</u></b>
III	Her iki taraf da zorunludur.	Durum II'ye bakın veya <u>iki varlığın tek bir varlıkta birbirine ait olmadığından emin olmak için modelinizi gözden geçirmeyi düşünün.</u>

# Departman ve Çalışan Arasındaki 1:1 İlişki

FIGURE 5.8 THE 1:1 RELATIONSHIP BETWEEN DEPARTMENT AND EMPLOYEE

A One-to-One (1:1) Relationship:

An EMPLOYEE manages zero or one DEPARTMENT;  
each DEPARTMENT is managed by one EMPLOYEE.





## Tasarım Örneği 2: Zaman Değişkeni Verilerinin Geçmişini Korumak

Şirket yöneticileri genellikle iyi karar vermenin veritabanlarında depolanan veriler aracılığıyla üretilen bilgilere dayandığını fark eder.

**Bu tür veriler hem **güncel** hem de **geçmiş** olayları yansıtır.**

**Zaman değişkenli veriler:** Değerleri zaman içinde değişen ve veri geçmişinin korunması gereken veriler.

## Tasarım Örneği 2: Maintaining History of Time-Variant Data

Örneğin; doğum tarihiniz veya Sosyal Güvenlik numaranız zaman değişkeni değildir.

Bir başka örnek; öğrenci not ortalamanız veya banka hesap bakiyeniz zaman içinde değişebilir.

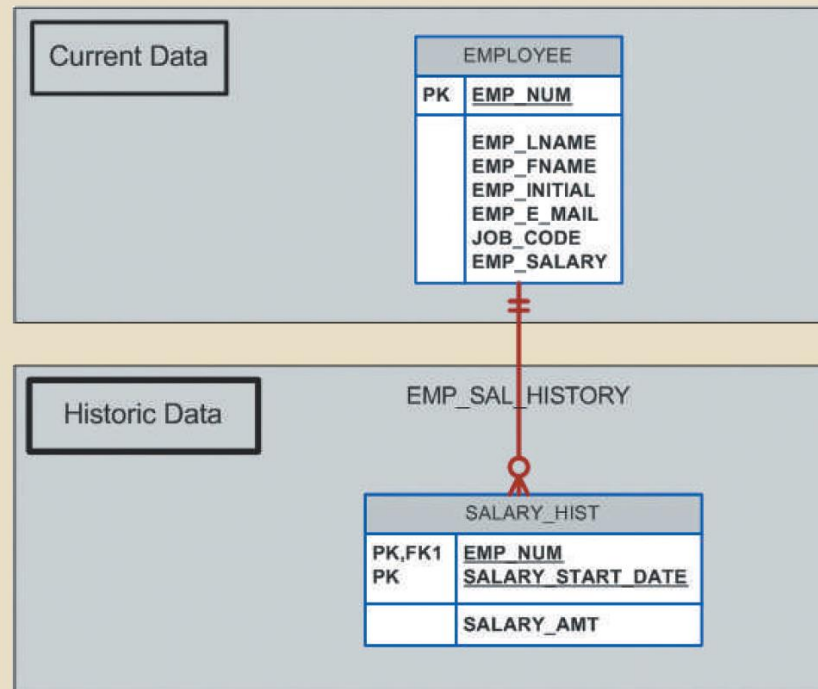
Bazen veri değişiklikleri dışarıdan kaynaklanır ve ürün fiyatı değişikliği gibi olay odaklıdır.

Zaman değişkenli verilerin depolanması, veri modelinde değişiklikler gerektirir; değişikliğin türü, verilerin yapısına bağlıdır.

**Örijinal varlıkla 1:M ilişkisinde yeni bir varlık oluşturulmasını gerektirir.**

Örneğin, her çalışan için maaş geçmişlerini izlemek istiyorsanız, aşağıdaki şekilde gösterildiği gibi EMP\_SALARY özniteliği çok değerli hale gelir.

FIGURE 5.9 MAINTAINING SALARY HISTORY



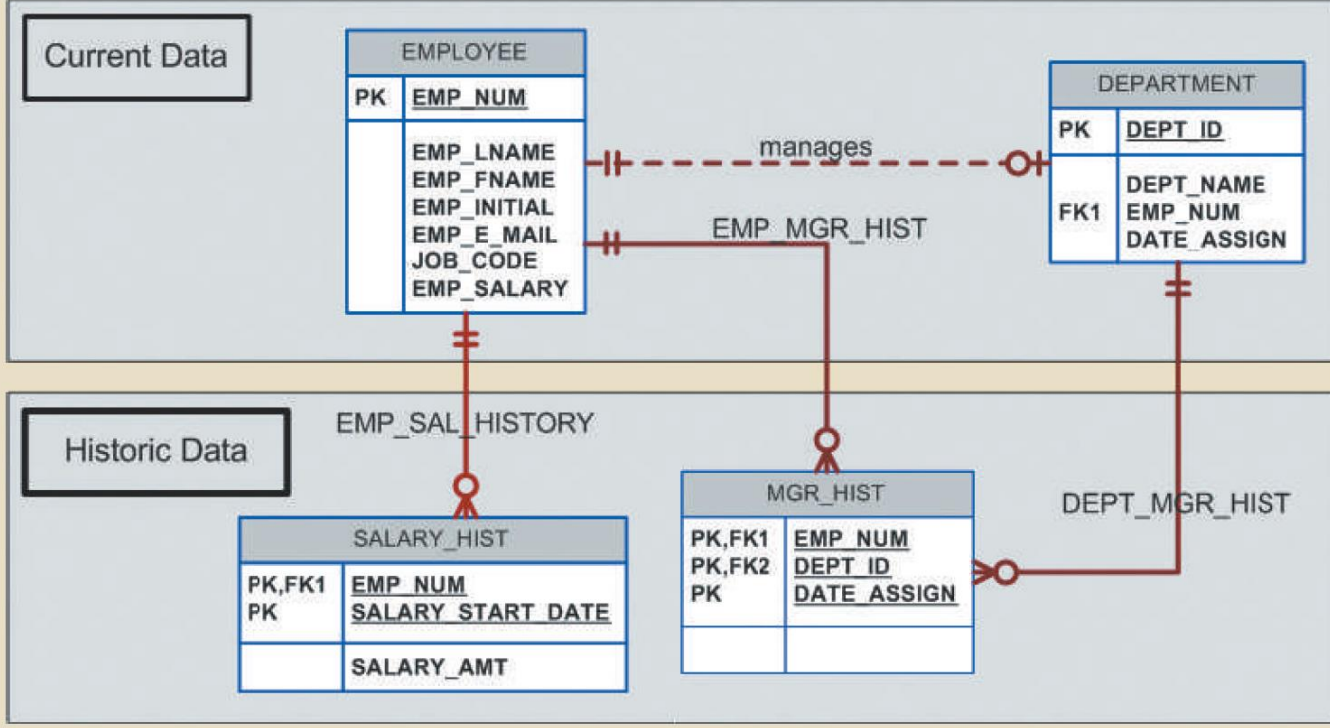
- **Diğer zaman değişkenli veriler, 1:M ilişkisini M:N ilişkisine dönüştürebilir.**
- Örneğin; veri modeliniz, **kuruluştaki farklı departmanlar ve her departmanı hangi çalışanın yönettiği** hakkındaki verileri içerir.
- **Bir departman için iş kuralları:**
  - Her departman sadece bir çalışan tarafından yönetilir.
  - Her çalışan bir departmanı yönetebilir.

Yani, **ÇALIŞAN ve DEPARTMAN** arasında 1: 1 ilişki olacaktır.

Bu ilişki, her departmanın mevcut yöneticisini kaydeder.

# Tüm departman yöneticilerinin geçmişini takip etmek istiyoruz.

FIGURE 5.10 MAINTAINING MANAGER HISTORY



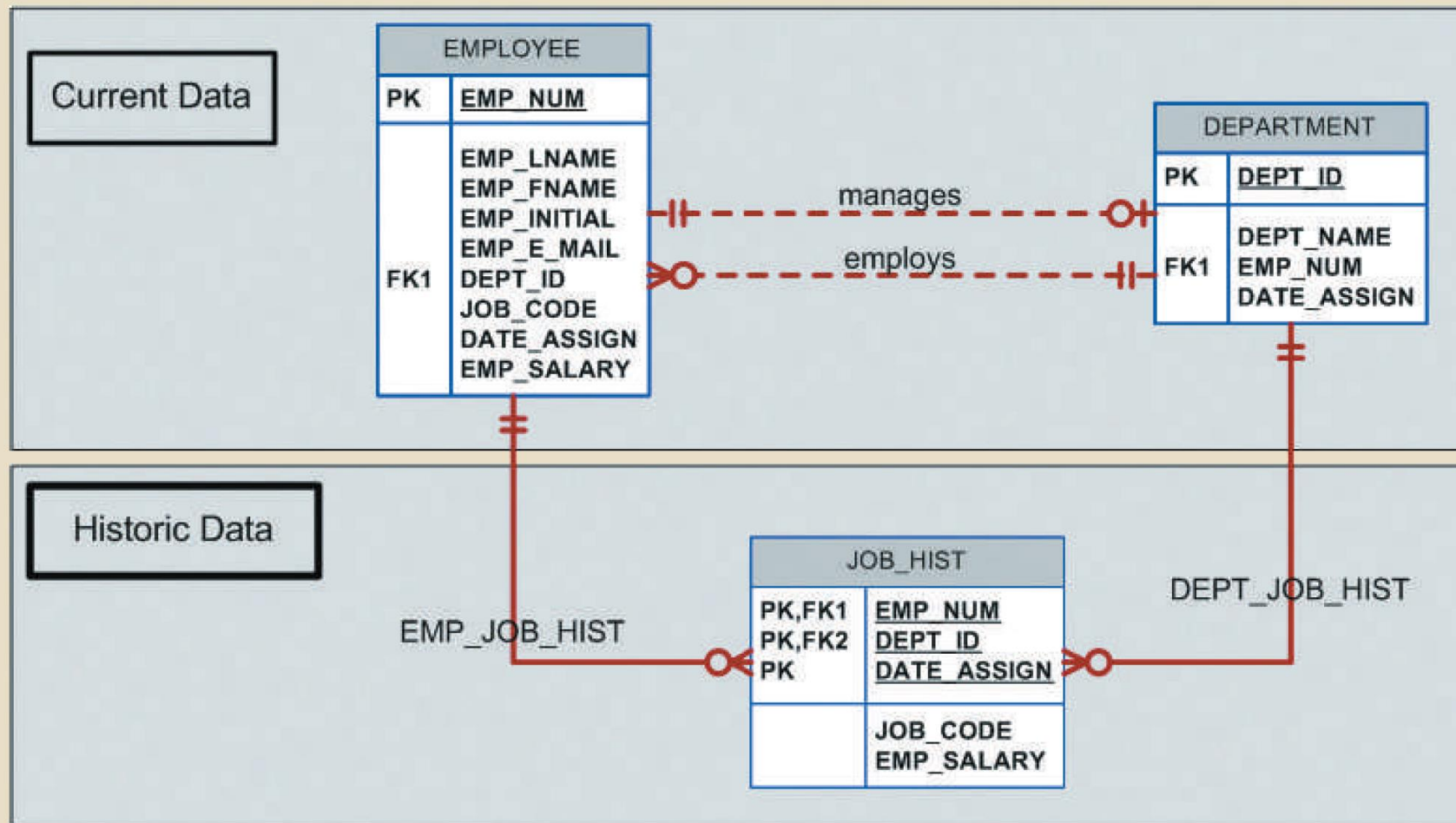
- Bu modelle yapılan takas, bir departmana her yeni bir yönetici atandığında, iki veri değişikliği olacaktır:
- DEPARTMENT varlığında bir güncelleştirme (date\_assing sütun). MGR\_HIST tablosuna bir satır eklenir.

Yukarıdaki şekilde "yönetir" ilişkisinin teoride isteğe bağlı ve pratikte fazlalık oluşturur.

İstediğiniz zaman, belirli bir departman için *MGR\_HIST* tablosundan *DATE\_ASSIGN* tarihini alarak bir departmanın yöneticisini tanımlayabilirsiniz.

- Şirket çalışanlarının her biri için iş geçmişini takip etmek istiyoruz.
- Bu görevi gerçekleştirmek için, JOB\_HIST bir varlık ekleyerek aşağıdaki şekildeki gibi modeli değiştirebilirsiniz.

■ FIGURE 5.11 MAINTAINING JOB HISTORY



## Tasarım Örneği 3: Fan Tuzakları

**Bir veri modeli oluşturmak, varlıklar arasındaki veri ilişkilerinin doğru bir şekilde tanımlanmasını gerektirir.**

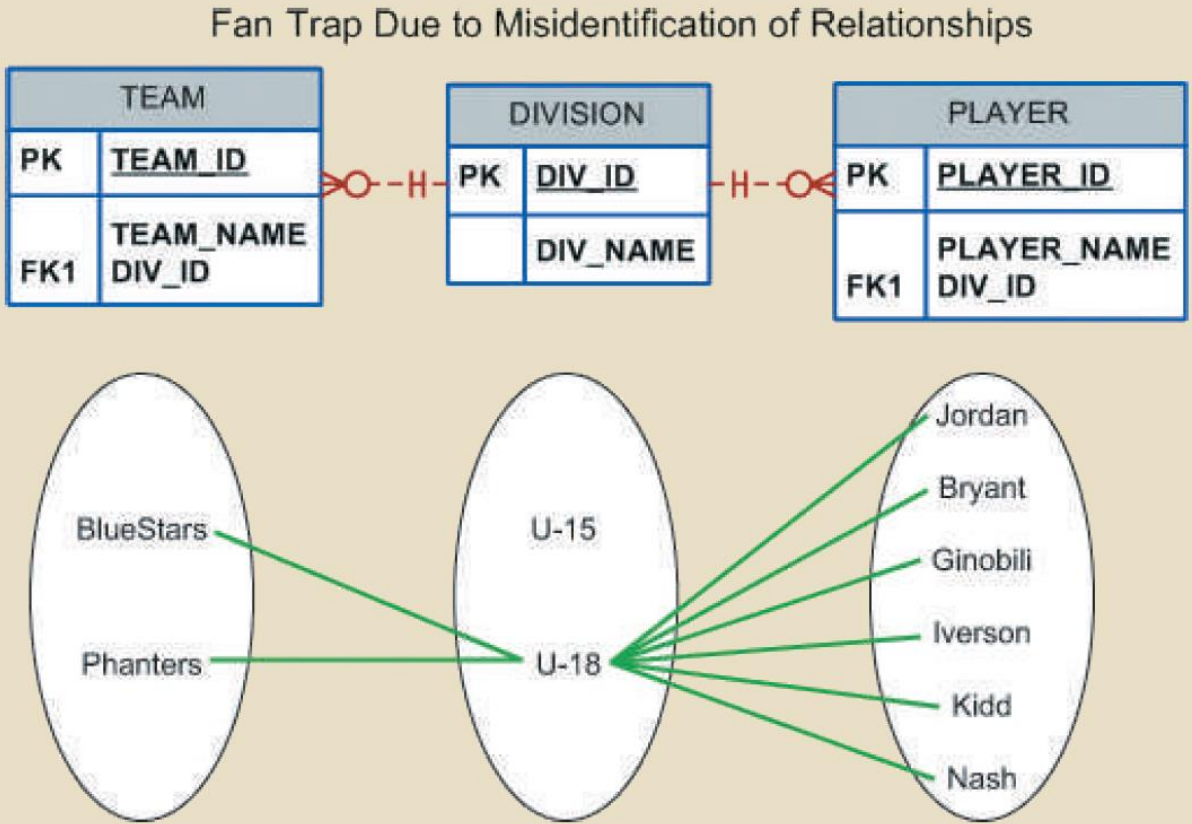
Bununla birlikte, iletişimsizlik veya iş kurallarının veya süreçlerinin eksik anlaşılması nedeniyle, varlıklar arasındaki ilişkileri yanlış tanımlanabilir.

## Tasarım Örneği 3: Fan Tuzakları

- **Tasarım tuzağı:** Bir ilişki yanlış veya eksik tanımlandığında oluşur.
  - En yaygın fan tasarım tuzağı olarak bilinir.
- **Fan tuzağı:** Bir varlık diğer varlıklarla **iki 1:M ilişkisinde olduğunda oluşur.**
  - Modelde ifade edilmeyen diğer varlıklar arasında bir ilişki üretir.



FIGURE 5.12 INCORRECT ERD WITH FAN TRAP PROBLEM



### Örneğin, iş kuralları:

- JCB basketbol liginin birçok bölümü vardır.
- Her klasmanın birçok oyuncusu vardır.
- Her klasmanın birçok takımı vardır.

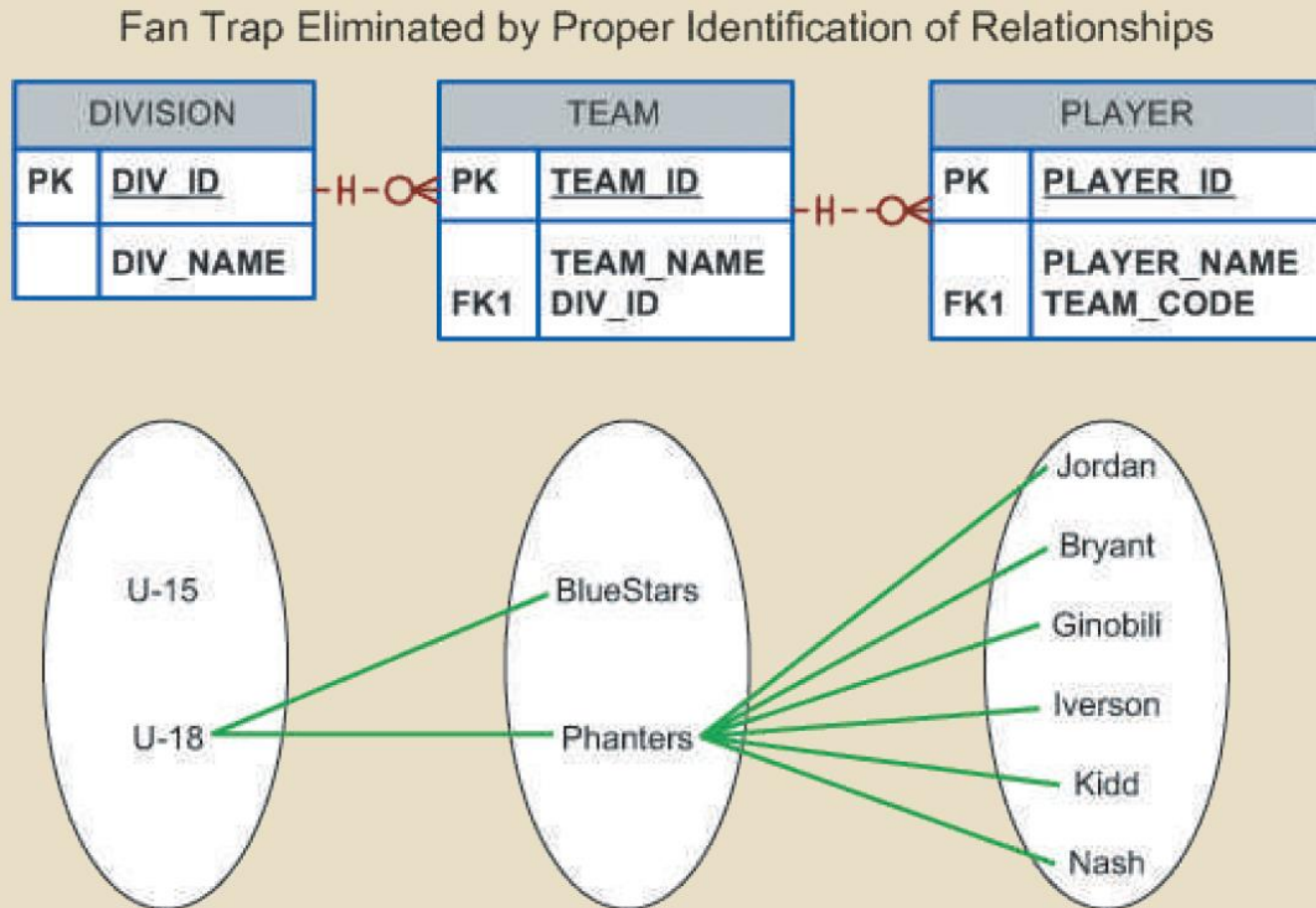
Bu "eksik" iş kuralları göz önüne alındığında, yandaki şekildekine benzeyen bir ERD oluşturabilirsiniz.

Bu temsil semantik olarak doğru olsa da, ilişkiler düzgün bir şekilde tanımlanmamıştır.

Örneğin, hangi oyuncuların hangi takıma ait olduğunu belirlemenin bir yolu yoktur.

- Fan tuzağını ortadan kaldırıldıktan sonra doğru ERD.

FIGURE 5.13 CORRECTED ERD AFTER REMOVAL OF THE FAN TRAP



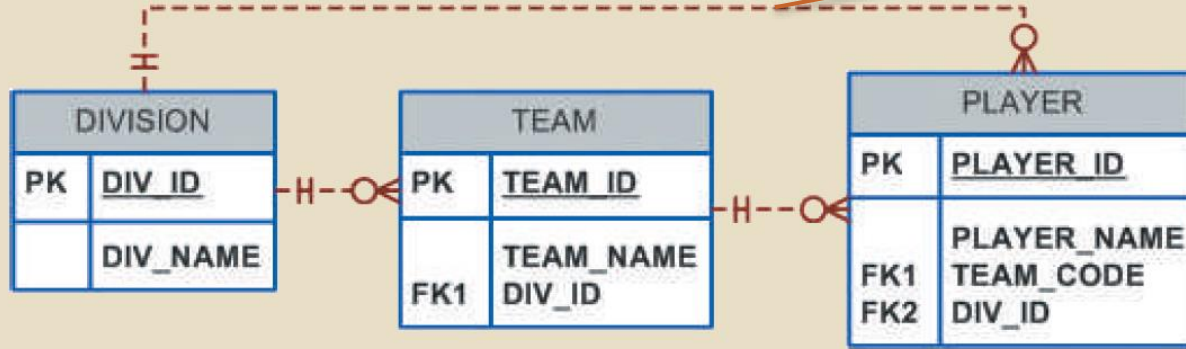
## Tasarım Örneği 4: Gereksiz İlişkiler

- Gereksiz ilişkiler, ilgili varlıklar arasında birden çok ilişki yolu olduğunda oluşur
- Fazlalıklar veri anormalliklerine ve veritabanında tutarsızlığa neden olabilir.
- Ancak, bazı tasarımların tasarımı, basitleştirmenin bir yolu olarak gereksiz ilişkiler kullandığını belirtmek önemlidir.
- Örneğin; EMPLOYEE ve DEPARTMENT arasındaki "yönetir" ve "istihdam eder" ilişkilerinin kullanımı gereksiz olarak görülmüştür. Fakat bu ilişkiler MGR\_HIST tablosunda tarihsel verilerden ziyade mevcut verilerle ilgilendiği gerçeğiyle gerekçelendirilmiştir.

- Gereksiz bir ilişkinin daha spesifik bir başka örneği aşağıda gösterilmiştir.

■

FIGURE 5.14 A REDUNDANT RELATIONSHIP



Redundant  
relation

Şekil 5.14'te, **TEAM** varlık kümesi aracılığıyla **DIVISION** ve **PLAYER** arasındaki geçişli **1:M** ilişkisine dikkat edin.

Bu nedenle, **DIVISION** ve **PLAYER**'ı birbirine bağlayan ilişki, tüm pratik amaçlar için gereksizdir.

Bu durumda, ilişki modeldeki herhangi bir bilgi oluşturma özelliğini kaybetmeden güvenli bir şekilde silinebilir.

## KAYNAKÇA

- Carlos Coronel, Steven Morris, DATABASE SYSTEMS, Design, Implementation, and Management, Cengage Learning, 13. edition