

### varlık süper tipi

Bir genelleme veya özelleştirme hiyerarşisinde, varlık alt türlerinin ortak özelliklerini içeren genel bir varlık türü.

**EMP\_RATINGS** (EMP\_RATINGS) ve **EMP\_MED\_TYPE** (EMP\_MED\_TYPE), pilot olmayan çalışanlar için boş değerler üretecektir. Buna ek olarak, pilotlar niteliklerine özgü bazı ilişkilere katılırlar. Örneğin, tüm çalışanlar uçak uçuramaz; sadece pilot olan çalışanlar "**çalışan uçak uçurur**" (employee flies airplane) ilişkisine katılabilir.

Önceki tartışmaya dayanarak, **PILOT** (PILOT) varlığının yalnızca pilotlara özgü nitelikleri depoladığını ve **EMPLOYEE** (EMPLOYEE) varlığının tüm çalışanlar için ortak olan nitelikleri depoladığını doğru bir şekilde çıkarırsınız. Bu hiyerarşiye dayanarak, **PILOT** (PILOT)'un **EMPLOYEE** (EMPLOYEE)'nin bir alt türü olduğu ve **EMPLOYEE** (EMPLOYEE)'nin **PILOT** (PILOT)'un süper türü olduğu sonucuna varabilirsiniz.

Modelleme açısından, bir **varlık üst tipi (entity supertype)**, bir veya daha fazla **varlık alt tipi (entity subtype)** ile ilişkili olan genel bir varlık tipidir. **Varlık üst tipi (entity supertype)** ortak özellikler içerir ve **varlık alt tipleri (entity subtypes)** her biri kendi benzersiz özelliklerini içerir.

## Şekil 5.1 Benzersiz Öznitelikler Tarafından Oluşturulan Boşluklar

Veritabanı adı: 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-92
101	Lewis	Marcos		ATP	SEL/MEL/Instr/CFII	1	25-Apr-93
102	Vandam	Jean					20-Dec-97
103	Jones	Victoria	R				28-Aug-07
104	Lange	Edith		ATP	SEL/MEL/Instr	1	20-Oct-01
105	Williams	Gabriel	U	COM	SEL/MEL/Instr/CFI	2	08-Nov-01
106	Duzak	Mario		COM	SEL/MEL/Instr	2	05-Jan-08
107	Diante	Venite	L				02-Jul-01
108	Wiesenbach	Joni					18-Nov-99
109	Travis	Brett	T	COM	SEL/MEL/SES/Instr/CFII	1	14-Apr-05
110	Genkazi	Stan					01-Dec-07

### varlık alt türü

Bir genelleme veya özelleştirme hiyerarşisinde, bir varlık üst tipinin bir alt kümesi. Varlık üst tipi şunları içerir  
Ortak özellikler ve alt tipler her bir varlığın kendine özgü özelliklerini içerir.

İki kriter, tasarımcının alt tipleri ve üst tipleri ne zaman kullanacağını belirlemesine yardımcı olur:

- Kullanıcının ortamında varlığın farklı, tanımlanabilir türleri veya tipleri bulunmalıdır.
- Farklı örnek türleri veya tiplerinin her biri, o örnek türüne veya tipine özgü bir veya daha fazla sahip olmalıdır.

Yukarıdaki örnekte, pilotlar hem tanımlanabilir bir çalışan türü olma hem de diğer çalışanların sahip olmadığı benzersiz niteliklere sahip olma kriterlerini karşıladığından, **PILOT**'u **ÇALIŞAN**'ın bir alt türü olarak oluşturmak uygundur. Mekanikerlerin ve muhasebecilerin de sırasıyla kendilerine özgü niteliklere sahip olduğunu ve memurların sahip olmadığını varsayalım. Bu durumda, **MEKANİK** ve **MUHASEBECİ** de **ÇALIŞAN**'ın meşru alt türleri olacaktır çünkü bunlar tanımlanabilir çalışan türleridir ve benzersiz niteliklere sahiptir. **CLERK**, **EMPLOYEE**'nin kabul edilebilir bir alt türü olmayacaktır çünkü kriterlerden yalnızca birini karşılamaktadır - tanımlanabilir bir çalışan türüdür - ancak niteliklerin hiçbirini yalnızca memurlara özgü değildir. Bir sonraki bölümde, varlık üst tiplerinin ve alt tiplerinin bir uzmanlaşma hiyerarşisinde nasıl ilişkili olduğunu öğreneceksiniz.

### uzmanlaşma hiyerarşisi

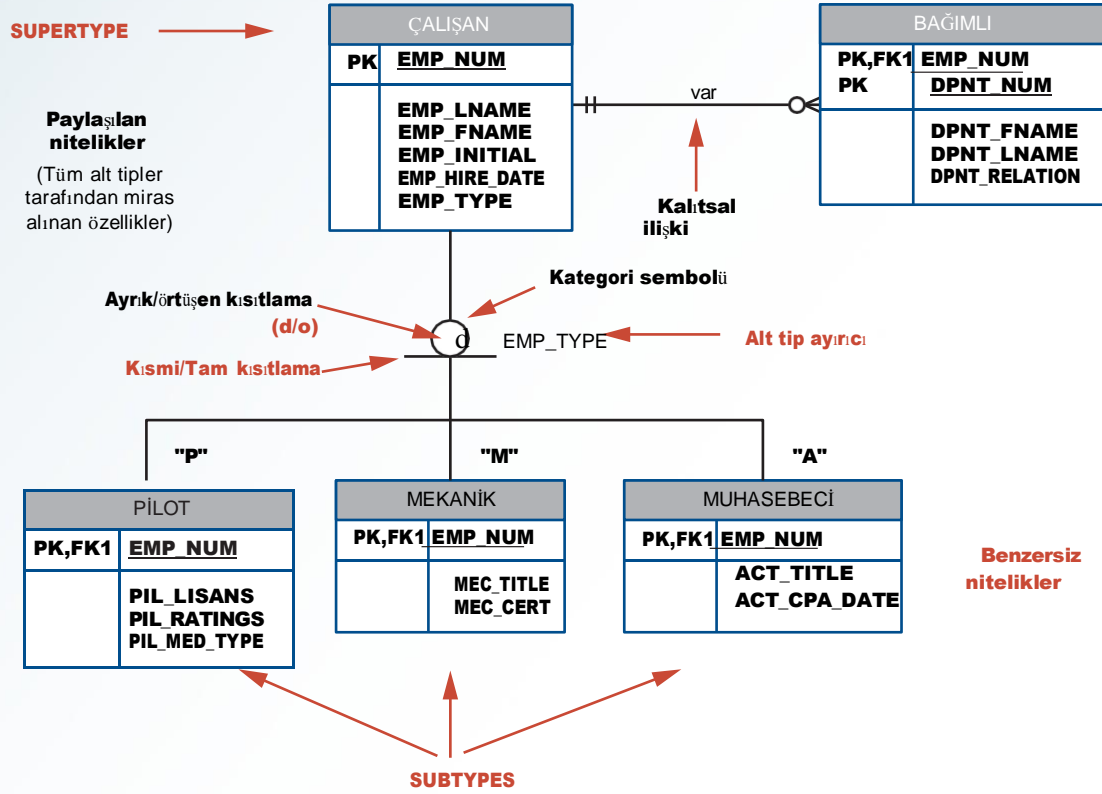
Yukarıdan aşağıya doğru bir hiyerarşi süreci daha düşük seviyeli, daha spesifik varlık alt tiplerinin tanımlanması daha üst düzey bir varlık süper tipinden. Özelleştirme, benzersiz özelliklerin gruplandırılmasına ve alt tiplerin ilişkileri.

## 5-1 b Uzmanlaşma Hiyerarşisi

Varlık üst tipleri ve alt tipleri, üst düzey varlık üst tiplerinin (ana varlıklar) ve alt düzey varlık alt tiplerinin (alt varlıklar) düzenini gösteren bir **uzmanlaşma hiyerarşisi** içinde düzenlenir. Şekil 5.2 bir **EMPLOYEE** üst tipi ve üç varlık alt tipi - **PILOT**, **MEKANİK** ve **MUHASEBECİ** - tarafından oluşturulan uzmanlaşma hiyerarşisini göstermektedir. Uzmanlaşma hiyerarşisi, **EMPLOYEE** ile alt türleri arasındaki 1:1 ilişkiyi yansıtmaktadır. Örneğin, bir **PILOT** alt tipi oluşumu **EMPLOYEE** üst tipinin bir örneğiyle ilişkilidir ve bir **MECHANIC** alt tipi oluşumu **EMPLOYEE** üst tipinin bir örneğiyle ilişkilidir. Şekil 5.2'deki terminoloji ve semboller bu bölüm boyunca açıklanmıştır.

Uzmanlık hiyerarşisi içinde tasvir edilen ilişkiler bazen "is-a" ilişkileri açısından tanımlanır. Örneğin, bir pilot bir çalışandır, bir teknisyen bir çalışandır ve bir

## Şekil 5.2 Bir Uzmanlaşma Hiyerarşisi



muhasebeci bir çalışandır. Bir uzmanlaşma hiyerarşisi içinde, bir alt tipin yalnızca bir üst tip bağlamında var olabileceğini ve her alt tipin doğrudan ilişkili olduğu yalnızca bir üst tipe sahip olabileceğini anlamak önemlidir. Bununla birlikte, bir uzmanlaşma hiyerarşisi birçok düzeyde üst tip veya alt tip ilişkisine sahip olabilir; yani, bir üst tipin birçok alt tipe sahip olduğu bir uzmanlaşma hiyerarşisine sahip olabilirsiniz. Buna karşılık, alt tiplerden biri diğer alt düzey alt tiplerin süper tipidir.

Şekil 5.2'de görebileceğiniz gibi, varlık üst tiplerinin ve alt tiplerinin bir özelleştirme hiyerarşisi içinde düzenlenmesi kozmetik bir kolaylıktan daha fazlasıdır. Uzmanlaşma hiyerarşileri, veri modelinin ERD'ye ek semantik içerik (anlam) yakalamasını sağlar. Bir uzmanlaşma hiyerarşisi aşağıdakiler için araçlar sağlar:

- Öznitelik kalıtımını destekleyin.
- Alt tip ayırıcı olarak bilinen özel bir üst tip niteliği tanımlayın.
- Ayrık veya örtüşen kısıtlamaları ve tam veya kısmi kısıtlamaları tanımlayın.

Aşağıdaki bölümler bu özellikleri ve kısıtlamaları daha ayrıntılı olarak ele almaktadır.

## 5-1c Kalıtım

**Kalıtım** özelliği, bir varlık alt tipinin üst tipin niteliklerini ve ilişkilerini miras almasını sağlar. Daha önce tartışıldığı gibi, bir süper tip tüm alt tipleri için ortak olan nitelikleri içerir. Buna karşın, alt tipler yalnızca alt tipe özgü nitelikleri içerir. Örneğin, Şekil 5.2 pilotların, teknisyenlerin ve muhasebecilerin hepsinin çalışan numarası, soyadı, adı, göbek adı ve işe alınma tarihini EMPLOYEE varlığından aldığını göstermektedir. Bununla birlikte, Şekil 5.2 pilotların benzersiz niteliklere sahip olduğunu da göstermektedir; aynı durum teknisyenler ve muhasebeciler için de geçerlidir. *Önemli bir kalıtım özelliği de tüm varlık alt türlerinin birincil anahtar niteliklerini üst türlerinden miras almasıdır.* Şekil 5.2'de EMP\_NUM niteliğinin her bir alt tür için birincil anahtar olduğuna dikkat edin.

## Çevrimiçi İçerik

Bu bölüm yalnızca uzmanlaşma hiyerarşilerini kapsamaktadır. EER modeli ayrıca bir alt tipin birden fazla ebeveyne (süper tip) sahip olabileceği özelleştirme kafeslerini de destekler. Ancak, bu kavramlar Ek G, Nesne Veritabanları'nda nesne yönelimli model altında daha iyi ele alınmıştır. Bu eke [www.cengage.com](http://www.cengage.com) adresinden ulaşılabilir.

## MİRAS

EERD'de, bir varlık alt tipinin varlık üst tipinin niteliklerini ve ilişkilerini miras sağlayan özellik

Uygulama düzeyinde, uzmanlaşma hiyerarşisinde gösterilen üst tip ve alt tipleri 1:1 ilişkisini korur. Örneğin, özelleştirme hiyerarşisi, Şekil 5.1'deki istenmeyen EMPLOYEE tablo yapısını, biri EMPLOYEE üst tipini ve diğeri PILOT alt tipini temsil eden iki tablo ile değiştirmenize olanak tanır. (Bkz. Şekil 5.3.)

### Şekil 5.3 Çalışan-Pilot Üst Tip-Alt Tip İlişkisi

Tablo adı: EMPLOYEE

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIRE_DATE	EMP_TYPE
100	Kolmycz	Xavier	T	15-Mar-92	
101	Lewis	Marcos		25-Apr-93	P
102	Vandam	Jean		20-Dec-97	A
103	Jones	Victoria	R	28-Aug-07	
104	Lange	Edith		20-Oct-01	P
105	Williams	Gabriel	U	08-Nov-01	P
106	Duzak	Mario		05-Jan-08	P
107	Diante	Venite	L	02-Jul-01	M
108	Wiesenbach	Joni		18-Nov-99	M
109	Travis	Brett	T	14-Apr-05	P
110	Genkazi	Stan		01-Dec-07	A

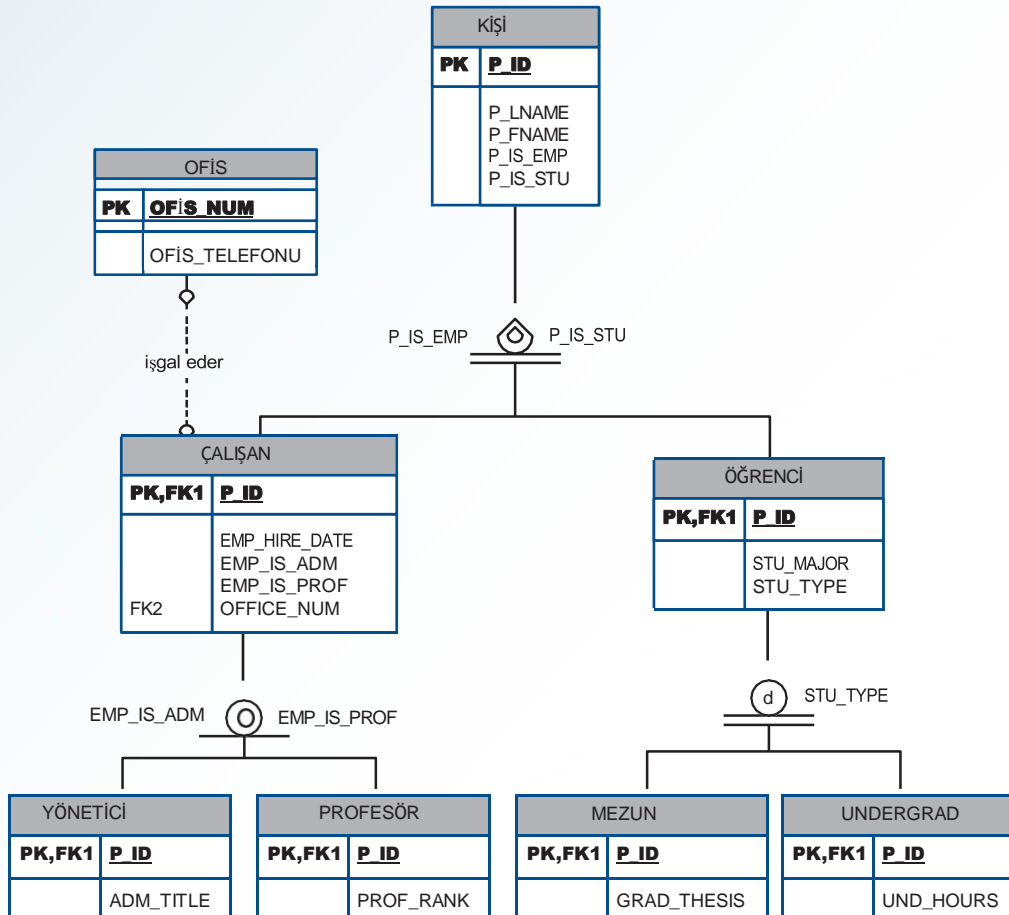
Veritabanı adı: Ch05\_AirCo

Tablo adı: PILOT

EMP_NUM	PIL_LICENSE	PIL_RATINGS	PIL_MED_TYPE
101	ATP	SEL/MEL/Instr/CFII	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/CFII	1

Varlık alt tipleri, üst tip varlığın katıldığı tüm ilişkileri devralır. Örneğin, Şekil 5.2, EMPLOYEE varlık üst tipinin DEPENDENT varlığı ile 1:M ilişkisine katıldığını göstermektedir. Kalıtım yoluyla, tüm alt tipler de bu ilişkiye katılır. Birden fazla üst tip ve alt tip seviyesine sahip uzmanlaşma hiyerarşilerinde, alt seviyedeki bir alt tip, üst seviyedeki tüm üst tiplerinden tüm nitelikleri ve ilişkileri miras alır.

### Şekil 5.4 Örtüşen Alt Tiplerle Uzmanlaşma Hiyerarşisi



Üst tiplerinin ilişkilerini miras almaları, alt tiplerin kendi ilişkilerine sahip olamayacakları anlamına gelmez. Şekil 5.4, PERSON'un bir alt türü olan EMPLOYEE ile OFFICE arasındaki 1:M ilişkisini göstermektedir. Bu sistemde yalnızca çalışanların ve başka hiçbir kişi türünün ofisi olmayacağı için, ilişki doğrudan alt tür ile modellenmiştir.

## 5-1d Alt Tip Ayırıcı

Bir **alt tip ayırıcı**, üst tip oluşumunun hangi alt tiplerle ilişkili olduğunu belirleyen üst tip varlığındaki özneliktir. Şekil 5.2'de, alt tip ayırıcı çalışan tipidir (EMP\_TYPE).

Şekil 5.2'de gösterildiği gibi, ERD'deki her alt tür için alt ayırıcısını ve değerini göstermek yaygın bir uygulamadır. Ancak, tüm ER modelleme araçları bu uygulamayı takip etmez. Şekil 5.2'de, ayırıcı değerini varlık alt türünün üzerine, bağlayıcı çizgisine yakın bir yere manuel olarak eklemek için bir metin aracı kullanılmıştır. Şekil 5.2'yi rehberiniz olarak kullanarak, EMP\_TYPE "P" değerine sahipse üst tipin bir PILOT alt tipiyle ilişkili olduğuna dikkat edin. EMP\_TYPE değeri "M" ise, üst tip bir MEKANİK alt tipiyle ilişkilidir. EMP\_TYPE değeri "A" ise, üst tip MUHASEBECİ alt tipiyle ilişkilidir.

Alt tür ayırıcı niteliği için varsayılan karşılaştırma koşulunun eşitlik karşılaştırması olduğunu unutmayın. Ancak, bazı durumlarda alt tür ayırıcısının eşitlik karşılaştırmasına dayanması gerekmez. Örneğin, iş gereksinimlerine bağlı olarak iki yeni pilot alt türü oluşturabilirsiniz: sorumlu pilot (PIC)-kalifiye ve yalnızca copilot-kalifiye. PIC nitelikli bir pilotun 1.500'den fazla PIC uçuş saatine sahip olması gerekir. Bu durumda, alt tip ayırıcı "Uçuş\_Saatleri" olur ve kriterler sırasıyla > 1.500 veya <= 1.500 olur.

## Not

Veri modelleme araçlarının tüm sürümleri uzmanlaşma hiyerarşisini yerel olarak desteklemez. Idera'nın E/R Studio Data Architect ve Sybase'in Power Designer gibi profesyonel veri modelleme araçları uzmanlaşma hiyerarşilerini destekler, ancak bu araçlar genellikle tek bir kullanıcı lisansı için binlerce dolara mal olur.

## 5-1e Ayrık ve Örtüşen Kısıtlamalar

Bir varlık üst tipi, ayrık veya örtüşen varlık alt tiplerine sahip olabilir. Havacılık örneğinde, bir çalışan pilot, teknisyen veya muhasebeci olabilir. İş kurallarından birinin, bir çalışanın aynı anda birden fazla alt türe ait olamayacağını belirttiğini varsayalım; , bir aynı anda hem pilot hem de teknisyen olamaz. **Çakışmayan alt tipler** olarak da bilinen **ayrık alt** tipler, süper tip varlık kümesinin *benzersiz bir* alt kümesini içeren alt tiplerdir; başka bir deyişle, süper tipin her varlık örneği alt tiplerden yalnızca birinde görünebilir. Örneğin, Şekil 5.2'de pilot (alt tip) olan bir çalışan (üst tip) yalnızca PILOT alt tipinde görünebilir, diğer alt tiplerin hiçbirinde görünemez. Bir ERD'de, bu tür ayrık alt tipler kategori sembolünün içinde *d* harfi ile gösterilir.

Öte yandan, iş kuralı çalışanların birden fazla sınıflandırmaya sahip olabileceğini belirtiyorsa, EMPLOYEE üst tipi *örtüşen* iş sınıflandırması alt tipleri içerebilir. **Çakışan alt tipler**, süper tip varlık kümesinin benzersiz olmayan alt kümelerini içeren alt tiplerdir; yani, süper tipin her varlık örneği birden fazla alt tipte görünebilir. Örneğin, bir üniversite ortamında, bir kişi bir çalışan, bir öğrenci veya her ikisi de olabilir. Buna karşılık, bir çalışan bir profesör olabileceği gibi bir yönetici de olabilir. Bir çalışan aynı zamanda öğrenci de olabileceğinden, ÖĞRENCİ ve ÇALIŞAN, KİŞİ üst tipinin örtüşen alt tipleridir; tıpkı PROFESÖR ve YÖNETİCİ'nin ÇALIŞAN üst tipinin örtüşen alt tipleri olması gibi. Şekil 5.4, çakışan alt türleri kategori sembolünün içindeki *o* harfi ile göstermektedir.

## Çevrimiçi İçerik

Gelişmiş veri oluşturma hakkında bir eğitim için modelleri için Ek A'ya bakınız, Lucidchart ile Veritabanı Tasarımı: A Tutorial, [www.cengage.com](http://www.cengage.com) adresinde

**alt tip ayırıcı** Her bir üst tip oluşumunun hangi varlık alt tipiyle ilişkili olduğunu belirleyen üst tip varlığındaki özneliktir.

## ayrık alt tipler

Bir özelleştirme hiyerarşisinde, bunlar benzersiz ve örtüşmeyen alt tür varlık kümeleridir.

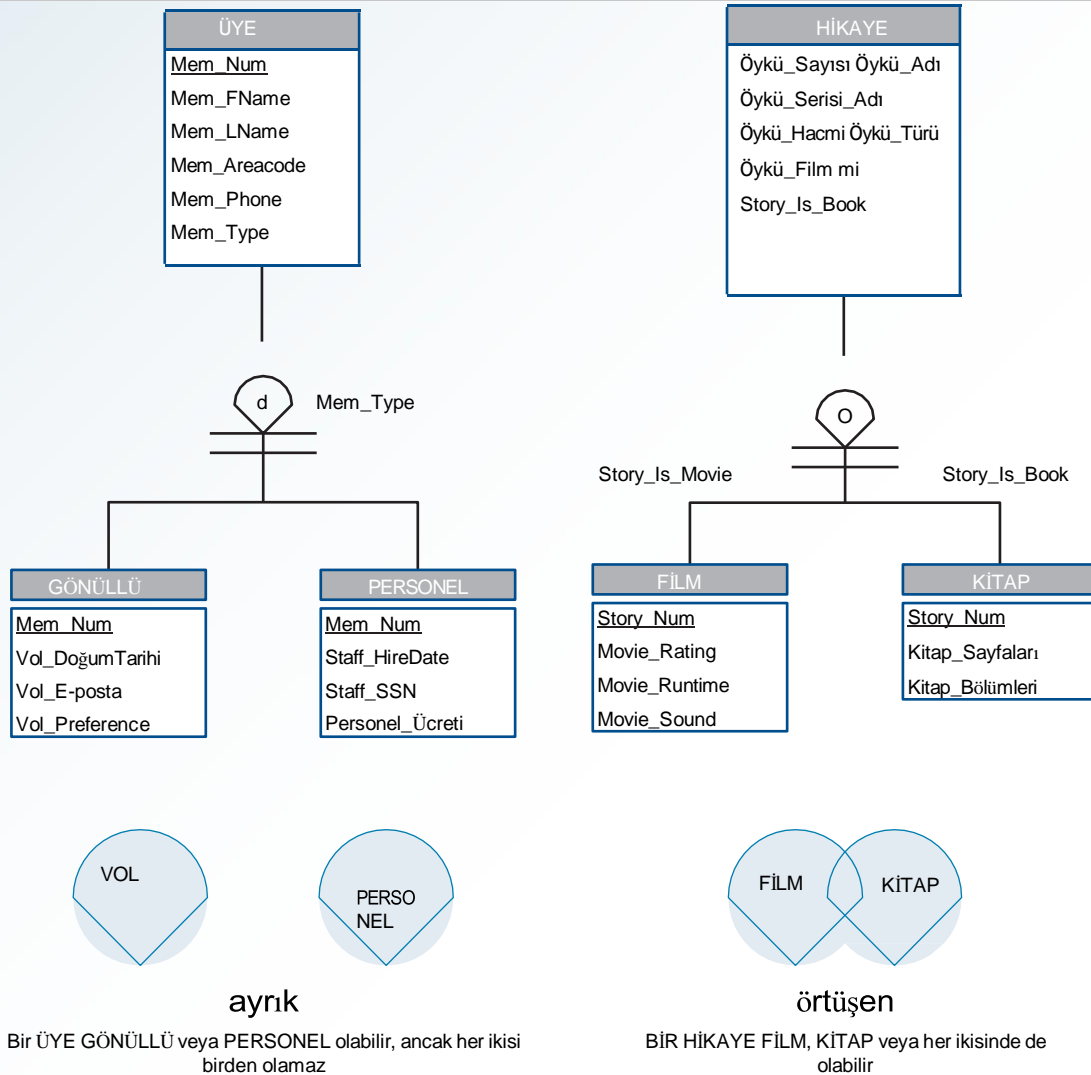
## örtüşmeyen alt tipler

Aynı alt tiplere bakınız.

**Örtüşen alt tipler** Bir uzmanlaşma hiyerarşisinde, üst tipin her bir varlık örneğinin (satır) birden fazla alt tipte görünebildiği bir durumdur.

Şekil 5.5, ayrık ve örtüşen alt tiplere ilişkin daha fazla örnek göstermektedir. Bir yardım kuruluşunda üyeler ya gönüllüdür ya da ücretli personeldir. Bir alt tipin, üst tipin tanımlanabilir bir türü veya örneği olması gerektiğini (bu örnekte, tanımlanabilir üye türleri) ve alt tipin bu tür veya örneğe özgü niteliklere sahip olması gerektiğini hatırlayın. Gönüllüler için geçerli olup ücretli personel için geçerli olmayan nitelikler vardır, bu nedenle GÖNÜLLÜ geçerli bir alt tiptir. Ücretli personel için geçerli olup gönüllüler için geçerli olmayan öz nitelikler vardır, bu nedenle STAFF da geçerli bir alt tiptir. Kuruluşun tek bir üyesi hem gönüllü hem de ücretli personel olamaz, bu nedenle bu alt tipler ayrıktır. Şekil 5.5'teki ikinci örnek, popüler çocuk hikayeleri hakkında veri sağlayan ve filmlerde ve kitaplarda yer alan hikayeleri takip eden bir web sitesidir. Filmlerdeki hikayeler, kitaplardaki hikayelerin sahip olmadığı derecelendirme ve ses türü gibi niteliklere sahiptir. Kitaplardaki hikayeler, bölüm ve sayfa sayısı gibi filmler için geçerli olmayan öz niteliklere sahiptir. FİLM ve KİTAP geçerli alt türlerdir çünkü her biri tanımlanabilir bir hikaye türüdür ve her biri diğer hikaye türü için geçerli olmayan niteliklere sahiptir. Ancak, bu alt türler örtüşmektedir çünkü bir hikayenin hem kitap hem de film olarak görünmesi mümkündür. Bu durumda, süper tip örneğinin eşleşen bir varlık alt tipine sahip olup olmadığını belirtmek için iki alt tip ayırt edici niteliğinin (Story\_Is\_Movie ve Story\_Is\_Book) kullanıldığına dikkat edin.

Şekil 5.5 Ayrık ve Örtüşen Alt Tipler



ERD'de ayrık ve örtüşen sembollerin gösterilmesi yaygın bir uygulamadır. (Bkz. Şekil 5.2, 5.4 ve 5.5.) Ancak, tüm ER modelleme araçları bu uygulamayı takip etmez. Örneğin, bazı araçlar bir alt tür ayrıcı sembolü sağlar, ancak ayrık ve örtüşen sembollerini sağlamaz. Bu durumda, önceki şekillerde  $d$  ve  $o$  sembollerini manuel olarak eklemek için bir metin aracı kullanılmıştır.

## Not

Ayrık ve örtüşen alt türleri temsil etmek için alternatif gösterimler mevcuttur. Örneğin, Toby J. Teorey ayrık ve örtüşen alt türleri göstermek için G ve Gs kullanımını popüler hale getirmiştir.

Bu bölümde daha önce öğrendiğiniz gibi, ayrık alt türlerin uygulanması, süper türdeki alt tür ayrıcı niteliğinin değerine dayanır. Ancak, üst üste binen alt tiplerin *uygulanması*, her alt tip için bir ayırt edici niteliğin kullanılmasını gerektirir. Örneğin, Bölüm 4, Varlık İlişkisi (ER) Modeling'deki Tiny College veritabanı tasarımı örneğinde, bir profesör aynı zamanda yönetici de olabilir. Bu nedenle, EMPLOYEE üst tipi Tablo 5.1'de gösterilen alt tip ayırt edici niteliklere ve değerlere sahip olacaktır.

**Tablo 5.1 Örtüşen Alt Tiplere Sahip Ayrıcı Öznelikler**

Ayrıcı Özellikler		Yorum
Profesör	Yönetici	
Y	N	Çalışan, Profesör alt türünün bir üyesidir.
N	Y	Çalışan, Yönetici alt türünün bir üyesidir.
Y	Y	Çalışan hem Profesör hem de İdarecidir.
N	N	Çalışanın herhangi bir alt türün üyesi olması gerekmez. Yalnızca hiyerarşi kısmi tamlık kısıtlaması sergiliyorsa mümkündür (sonraki bölüme bakın).

## 5-1f Tamlık Kısıtı

**Tamlık kısıtı**, her bir varlık üst tipi oluşumunun aynı zamanda en az bir alt tipin üyesi olması gerekerek gerekmediğini belirtir. Tamlık kısıtı kısmi veya toplam olabilir. **Kısmi tamlık**, her süper tip oluşumunun bir alt tipin üyesi olmadığı anlamına gelir; bazı süper tip oluşumları herhangi bir alt tipin üyesi olmayabilir. **Toplam tamlık**, her süper tip oluşumunun en az bir alt tipin üyesi olması gerektiği anlamına gelir.

Önceki şekillerdeki ERD'ler Kategori sembolüne dayalı tamlık kısıtını temsil etmektedir. Dairenin altındaki tek bir yatay çizgi kısmi tamlık kısıtlamasını; dairenin altındaki çift yatay çizgi ise tam tamlık kısıtlamasını temsil eder.

## Not

Tamlık kısıtlamasını temsil etmek için alternatif gösterimler mevcuttur. Örneğin, bazı gösterimler üst tipi Kategori sembolüne bağlamak için tek bir çizgi (kısmi) veya çift çizgi (toplam) kullanır.

Ayrık ve örtüşen alt tipler ve tamlık kısıtlamaları göz önüne alındığında, Tablo 5.2'de gösterilen uzmanlaşma hiyerarşisi kısıtlama senaryolarına sahip olmak mümkündür.

**Bütünlük** kısıtı Her bir varlık süper tipinin oluşup oluşmadığını belirten bir kısıt ayrıca en az bir alt türün üyesi olmalıdır. Alt tip tamlık kısıtı kısmi veya toplam .



**kısmi tamlık** Bir genelleme veya uzmanlaşma hiyerarşisinde, bazı durumlarda süper tip oluşumları herhangi bir alt tipin üyesi olmayabilir.

### toplam bütünlük

Bir genelleme veya özelleştirme hiyerarşisinde, her süper tip oluşumunun en az bir alt tipin üyesi olması gereken bir koşul.



**Tablo 5.2** Uzmanlaşma Hiyerarşisi Kısıtlama Senaryoları

Tip	Ayrık Kısıtlama	Çakışan Kısıtlama
Kısmi 	Supertype isteğe bağlı alt tiplere sahiptir. Alt tip ayrıncı null olabilir. Alt tip kümeleri benzersizdir.	Supertype isteğe bağlı alt tiplere sahiptir. Alt tip ayrıncı null olabilir. Alt tip kümeleri benzersiz değildir.
Toplam 	Her süper tip oluşumu yalnızca bir alt tipin üyesidir. Alt tip ayrıncı null olamaz. Alt tür kümeleri benzersizdir.	Her süper tip oluşumu en az bir alt tipin üyesidir. Alt tip ayrıncıları null olamaz. Alt tür kümeleri benzersiz değildir.

**uzmanlaşma**

Bir özelleştirme hiyerarşisinde, benzersiz niteliklerin bir alt tip varlık olarak gruplandırılması.

**genelleme**

Bir özelleştirme hiyerarşisinde, ortak niteliklerin bir süper tip varlık içinde gruplandırılması.

## 5-1g Uzmanlaşma ve Genelleştirme

Varlık üst tipleri ve alt tipleri geliştirmek için çeşitli yaklaşımlar kullanabilirsiniz. Örneğin, önce normal bir varlık tanımlayabilir ve ardından ayırt edici özelliklerine göre tüm varlık alt türlerini belirleyebilirsiniz. Ayrıca birden fazla varlık türü tanımlayarak başlayabilir ve daha sonra bu varlıkların ortak özelliklerini çıkararak daha üst düzey bir süper tip varlık oluşturabilirsiniz.

**Uzmanlaştırma**, daha üst düzey bir varlık üst tipinden daha alt düzey, daha spesifik varlık alt tiplerinin tanımlanmasına yönelik yukarıdan aşağıya bir süreçtir. Uzmanlaştırma, alt tiplerin benzersiz özelliklerinin ve ilişkilerinin gruplandırılmasına dayanır. Havacılık örneğinde, orijinal çalışan üst tipinden birden fazla varlık alt tipi tanımlamak için uzmanlaşmayı kullandınız. **Genelleştirme**, alt düzey varlık alt tiplerinden daha üst düzey, daha genel bir varlık üst tipi tanımlamaya yönelik aşağıdan yukarıya bir süreçtir. Genelleştirme, alt tiplerin ortak özelliklerinin ve ilişkilerinin gruplandırılmasına dayanır. Örneğin, birden fazla müzik aleti türü tanımlayabilirsiniz: piyano, keman ve gitar. Genelleştirme yaklaşımını kullanarak, birden fazla alt türün ortak özelliklerini tutmak için bir "yaylı çalgı" varlık üst türü tanımlayabilirsiniz.

## 5-2 Varlık Kümeleme

Bir ER diyagramı geliştirmek, muhtemelen yüzlerce varlık türünün ve bunların ilgili ilişkilerinin keşfedilmesini gerektirir. Genel olarak, veri modelleyici birkaç varlık içeren başlangıç ERD'si geliştirecektir. Tasarım tamamlanmaya yaklaştıkça, ERD, diyagramı okunamaz ve bir iletişim aracı olarak verimsiz hale getirecek kadar kalabalıklaştıran yüzlerce varlık ve ilişki içerecektir. Bu gibi durumlarda, ERD'de gösterilen varlıkların sayısını en aza indirmek için varlık kümelerini kullanabilirsiniz.

**Varlık kümesi**, ERD'de birden fazla varlığı ve ilişkiyi temsil etmek için kullanılan "sanal" bir varlık türüdür. Bir varlık kümesi, birbiriyle ilişkili birden fazla varlığın tek bir soyut varlık nesnesinde oluşturulur. Bir varlık kümesi, nihai ERD'de gerçekte bir varlık olmaması 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 "kavramsal" bir varlıktır.

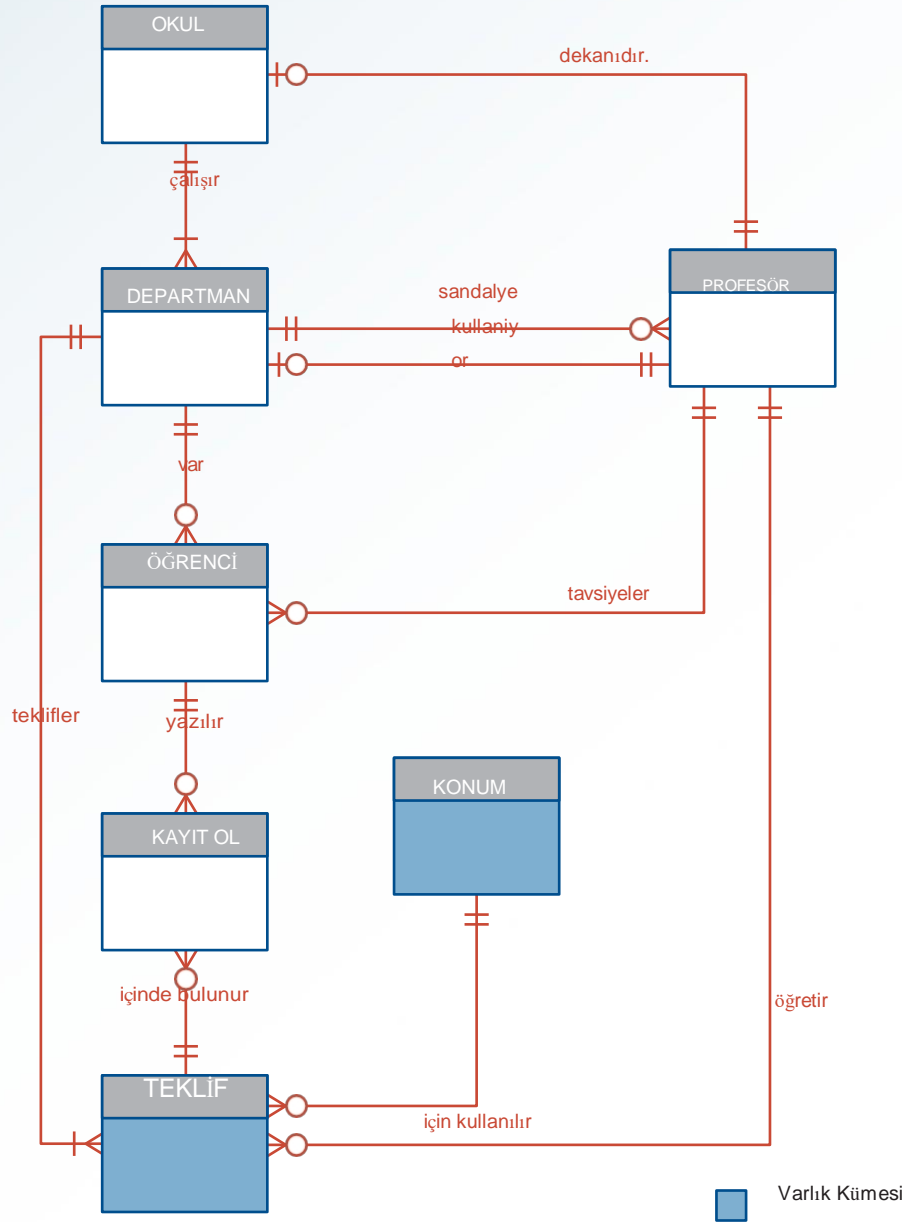
Şekil 5.6, Bölüm 4'teki Tiny College örneğine dayalı olarak varlık kümelerinin kullanımını göstermektedir. ERD'nin iki varlık kümesi içerdiğine dikkat edin:

- SEMESTER, KURS ve SINIF varlıklarını ve ilişkilerini gruplayan OFFERING
- ODA ve BİNA varlıklarını ve ilişkilerini gruplayan KONUM

**varlık kümesi**

Birden fazla varlığı temsil etmek için kullanılan "sanal" varlık türü ERD'deki varlıklar ve ilişkiler. Bir varlık kümesi birbiriyle ilişkili birden fazla varlığın tek bir soyut varlık nesnesinde oluşturulur. Bir varlık kümesi "sanal" veya "soyut" olarak kabul edilir çünkü nihai ERD'de gerçekte bir varlık değildir.

Şekil 5.6 Varlık Kümelerini Kullanan Küçük Üniversite ERD'si



Şekil 5.6'daki ERD'nin varlıklar için öznitelikleri göstermediğine de dikkat edin. Varlık kümeleri kullanıldığında, birleştirilmiş varlıkların anahtar nitelikleri artık mevcut değildir. Anahtar nitelikler olmadan, birincil anahtar kalıtım kuralları değişir. Buna karşılık, kalıtım kurallarındaki değişiklik, ilişkilerde değişiklikler (tanımlayıcıdan tanımlayıcı olmayana veya tam tersi) ve bazı varlıkların yabancı anahtar niteliklerinin kaybı gibi istenmeyen sonuçlara yol açabilir. Bu sorunları ortadan kaldırmak için genel kural, *varlık kümeleri kullanıldığında özniteliklerin görüntülenmesinden kaçınılmaktır*.

### 5-3 Varlık Bütünlüğü: Birincil Anahtarları Seçme

Bir varlığın tartışmasız en önemli özelliği, her bir varlık örneğini benzersiz bir şekilde tanımlayan birincil anahtardır (tek bir nitelik veya bazı nitelik kombinasyonları). Birincil anahtarın işlevi varlık bütünlüğünü garanti etmektir. Ayrıca, birincil anahtarlar ve yabancı anahtarlar



### doğal anahtar (doğal tanımlayıcı)

Gerçek dünya nesneleri için genel kabul görmüş bir tanımlayıcı. Adından anlaşılacağı gibi, doğal bir anahtar son kullanıcılar için tanıdık ve günlük iş kelime dağarcıklarının bir parçasını oluşturur.

ilişkisel modelde ilişkileri uygulamak için birlikte çalışır. Bu nedenle, birincil anahtarın doğru seçilmesinin önemi, veritabanı uygulamasının verimliliği ve etkinliği üzerinde doğrudan bir etkiye sahiptir.

## 5-3a Doğal Anahtarlar ve Birincil Anahtarlar

Benzersiz tanımlayıcı kavramına gerçek dünyada sıkça rastlanır. Örneğin, derslere kaydolmak için sınıf veya bölüm numaralarını, belirli faturaları tanımlamak için fatura numaralarını ve kredi kartlarını tanımlamak için hesap numaralarını kullanırsınız. Bu örnekler doğal tanımlayıcıları veya anahtarları göstermektedir. **Doğal anahtar** veya **doğal tanımlayıcı**, gerçek dünyadaki nesneleri ayırt etmek, yani benzersiz bir şekilde tanımlamak için kullanılan, gerçek dünyada genel kabul görmüş bir tanımlayıcıdır. Adından da anlaşılacağı üzere, bir doğal anahtar son kullanıcılar için tanıdık ve günlük iş kelime dağarcıklarının bir parçasını oluşturur.

Genellikle, bir varlığın doğal bir tanımlayıcısı varsa, veri modelleyici bunu modellenen *varlığın* birincil anahtarı olarak kullanır. Genel olarak, çoğu doğal anahtar kabul edilebilir birincil anahtar tanımlayıcıları oluşturur. Bir sonraki bölümde, birincil anahtarların seçilmesine ilişkin bazı temel yönergeler sunulmaktadır.

## 5-3b Birincil Anahtar Yönergeleri

Birincil anahtar, bir varlık kümesindeki varlık örneklerini benzersiz bir şekilde tanımlayan öznelik veya öznelik kombinasyonudur. Ancak, birincil anahtar örneğin 12 özneliğe dayalı olabilir mi? Ve bir birincil anahtar ne kadar uzun olabilir? Önceki örneklerde, EMP\_NUM neden EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL ve EMP\_DOB'un bir kombinasyonu değil de EMPLOYEE'nin birincil anahtarı olarak seçilmiştir? 256 baytlık tek bir metin özneliği iyi bir birincil anahtar olabilir mi? Bu soruların tek bir cevabı yoktur, ancak veritabanı uzmanları yıllar içinde bir uygulama bütünü oluşturmuştur. Bu bölüm, belgelenmiş uygulamalar bütünüdür.

Öncelikle, birincil anahtarın işlevini anlamalısınız. Ana işlevi, bir tablo içindeki bir varlık örneğini veya satırı benzersiz bir şekilde tanımlamaktır. Özellikle, bir birincil anahtar değeri -yani belirleyici- verildiğinde, ilişkisel model varlığı "tanımlayan" tüm bağımlı niteliklerin değerini belirleyebilir. Tanımlama ve açıklamanın modelde ayrı anlamsal yapılar olduğunu unutmayın. *Birincil anahtarın işlevi, varlığı "tanımlamak" değil, varlık bütünlüğünü garanti etmektir.*

İkinci olarak, varlıklar arasındaki ilişkileri uygulamak için birincil anahtarlar ve yabancı anahtarlar kullanılır. Ancak, bu tür ilişkilerin uygulanması çoğunlukla perde arkasında, son kullanıcılardan gizlenerek yapılır. Gerçek dünyada son kullanıcılar nesneleri, nesneler hakkında bildikleri özelliklere göre tanımlarlar. Örneğin, bir markette alışveriş yaparken, ürünleri stok numarasına bakarak değil, teşhir rafından alıp etiketlerini okuyarak seçersiniz. Veritabanı uygulamalarının insan seçim sürecini mümkün olduğunca taklit etmesi akıllıca olacaktır. Bu nedenle, veritabanı uygulamaları, perde arkasında birincil anahtar değerlerini kullanırken, son kullanıcının farklı nesnelerin çoklu tanımlayıcı anlatımları arasında seçim yapmasına izin vermelidir. Bu kavramları aklınızda tutarak, arzu edilen birincil anahtar özelliklerini özetleyen Tablo 5.3'e bakın.

## 5-3c Bileşik Birincil Anahtarlar Ne Zaman Kullanılır?

Önceki bölümde, birincil anahtarların istenen özelliklerini öğrendiniz. Örneğin, birincil anahtarın mümkün olan en az sayıda öznelik kullanması gerektiğini öğrendiniz. Ancak bu, bir modelde bileşik birincil anahtarlar için verilmediği anlamına gelmez. Aslında, bileşik birincil anahtarlar özellikle iki durumda kullanışlıdır:

- Her bir birincil anahtar kombinasyonuna M:N ilişkisinde yalnızca bir kez izin verilen bileşik varlıkların tanımlayıcıları olarak
- Zayıf varlığın ana varlıkla güçlü bir tanımlama ilişkisine sahip olduğu zayıf varlıkların tanımlayıcıları olarak

**Tablo 5.3** Arzu Edilen Birincil Anahtar Özellikler

PK Karakteristiği	Gerekçe
Benzersiz değerler	PK, her bir varlık örneğini benzersiz bir şekilde tanımlamalıdır. Bir birincil anahtar benzersiz değerleri garanti edebilmelidir. Boş değerler içeremez.
Akıllı olmayan	PK, her bir varlık örneğini benzersiz bir şekilde tanımlamak dışında gömülü anlamsal bir anlama sahip olmamalıdır. Gömülü anlamsal anlamı olan bir öznelik muhtemelen bir tanımlayıcıdan ziyade varlığın açıklayıcı bir özelliği olarak daha iyi kullanılır. Örneğin, birincil anahtar tanımlayıcısı olarak Smith, Martha L. yerine 650973 öğrenci kimliği tercih edilir.
Zaman içinde değişiklik yok	Bir niteliğin anlamsal bir anlamı varsa, güncellemelere tabi olabilir, bu nedenle isimler iyi birincil anahtarlar değildir. Vickie Smith birincil anahtarsa, evlendiğinde adını değiştirirse ne olur? Bir anahtar değişime tabi ise, yabancı anahtar değerlerinin de güncellenmesi gerekir, bu da veritabanı iş yükünü artırır. Ayrıca, bir birincil anahtar değerini değiştirmek, temelde bir varlığın kimliğini değiştirdiğiniz anlamına gelir. Kısacası, PK kalıcı ve değiştirilemez olmalıdır.
Tercihen tek özellikli	Bir birincil anahtar mümkün olan en az sayıda özneliğe sahip olmalıdır (indirgenemez). Tek öznelikli birincil anahtarlar arzu edilir ancak gerekli değildir. Tek öznelikli birincil anahtarlar yabancı anahtarların uygulanmasını basitleştirir. Çok birincil anahtarlara sahip olmak, ilgili varlıkların birincil anahtarlarının birçok öznelik eklenerek büyümesine neden olabilir, böylece veritabanı iş yükünü artırır ve (uygulama) kodlamayı daha zahmetli hale getirir.
Tercihen sayısal	Benzersiz değerler sayısal olduklarında daha iyi yönetilebilirler çünkü veritabanı, her yeni satırın eklenmesiyle değerleri otomatik olarak artıran sayaç tarzı bir öznelik uygulamak için dahili rutinleri kullanabilir. Aslında çoğu veritabanı sistemi, kendi kendini artıran birincil anahtar niteliklerini desteklemek için Microsoft Access'teki AutoNumber, Oracle'daki sequence, MySQL'deki autoincrement constraint veya MS SQL Server'daki identity özelliği gibi özel yapıları kullanma becerisini içerir.
Güvenlik uyumlu	Seçilen birincil anahtar, güvenlik riski veya ihlali olarak değerlendirilebilecek herhangi bir öznelikten/özneliklerden oluşmamalıdır. Örneğin, bir EMPLOYEE tablosunda PK olarak Sosyal Güvenlik numarası kullanmak iyi bir fikir değildir.

İlk durumu açıklamak için, bir STUDENT varlık kümeniz ve bir CLASS varlık kümeniz olduğunu varsayın. Ayrıca, bu iki kümenin bir ENROLL varlık kümesi aracılığıyla bir M:N ilişkisi içinde ilişkili olduğunu ve her öğrenci veya sınıf kombinasyonunun bileşik varlıkta yalnızca bir kez görünebileceğini varsayın. Şekil 5.7 böyle bir ilişkiyi temsil eden ERD'yi göstermektedir.

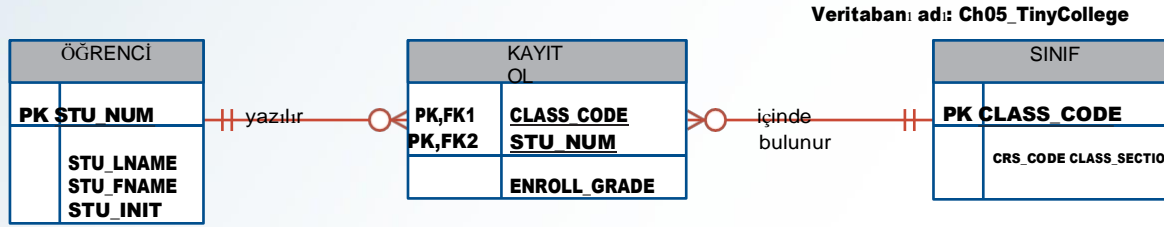
Şekil 5.7'de gösterildiği gibi, bileşik birincil anahtar otomatik olarak yinelenen değerlerin olmamasını sağlama avantajı sağlar; yani aynı öğrencinin aynı sınıfa birden fazla kez kaydolmamasını sağlar.

İkinci durumda, bir ana varlık ile güçlü bir tanımlama ilişkisi içinde olan zayıf bir varlık normalde iki durumdan birini temsil etmek için kullanılır:

1. *Varlığı başka bir gerçek dünya nesnesine bağlı olan bir gerçek dünya nesnesi.* Bu tür nesneler gerçek dünyada ayırt edilebilir. Bir bağımlı ve bir çalışan birbirinden bağımsız olarak var olan iki ayrı kişidir. Ancak, bu tür nesneler modelde ancak birbirleriyle güçlü bir tanımlayıcı ilişki içinde olduklarında var olabilirler. Örneğin, EMPLOYEE ve DEPENDENT arasındaki ilişki, bağımlı varlığın birincil anahtarının ana varlığın anahtarını içeren bir bileşik anahtar olduğu bir varlık bağımlılığı ilişkisidir.
2. *Veri modelinde güçlü bir tanımlama ilişkisi içinde iki ayrı varlık olarak temsil edilen bir gerçek dünya nesnesi.* Örneğin, gerçek dünyadaki fatura nesnesi bir veri modelinde iki varlık tarafından temsil edilir: INVOICE ve LINE. LINE varlığının gerçek dünyada bağımsız bir nesne olarak değil, INVOICE'ın bir parçası olarak var olduğu açıktır.

Her iki durumda da, güçlü bir tanımlama ilişkisine sahip olmak, bağımlı varlığın yalnızca ana varlıkla ilişkili olduğunda var olabilmesini sağlar. Özet olarak, bileşik ve zayıf varlık türleri için bileşik birincil anahtar seçimi modelin bütünlüğünü ve tutarlılığını artıran faydalar sağlar.

## Şekil 5.7 ÖĞRENCİ ve SINIF Arasındaki M:N İlişkisi



Tablo adı: ÖĞRENCİ (ilk dört alan)

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

Tablo adı: 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

Tablo adı: CLASS (ilk üç alan)

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

### 5-3d Vekil Birincil Anahtarlar Ne Zaman Kullanılır?

#### vekil anahtar

Sistem tarafından atanan, genellikle sayısal ve otomatik olarak artırılan birincil anahtar.

Bazı durumlarda, gerçek dünyada bir birincil anahtar bulunmayabilir veya mevcut doğal anahtar uygun bir birincil olmayabilir. Bu durumlarda, bir vekil anahtar oluşturmak standart bir uygulamadır. **Vekil** anahtar, varlık örneklerinin tanımlanmasını basitleştirmek için veritabanı tasarımcısı tarafından oluşturulan bir birincil anahtardır. Vekil anahtarın kullanıcı ortamında hiçbir anlamı yoktur; yalnızca bir varlık örneğini diğerinden ayırt etmek için vardır (tıpkı diğer birincil anahtarlar gibi). Vekil anahtarın pratik bir avantajı, kendine özgü bir anlamı olmadığından, benzersiz değerlerin her zaman sağlandığından emin olmak için VTYS tarafından değerlerin oluşturulabilmesidir.

Örneğin, küçük partiler için oda kiralayan bir park rekreasyon tesisini düşünün. Tesisin yöneticisi, Tablo 5.4'te gösterilen formatta bir klasör kullanarak tüm etkinliklerin kaydını tutar.

Tablo 5.4 Olayları Takip Etmek İçin Kullanılan Veriler

Tarih	Time_Start	Zaman_Sonu	Oda	Event_Name	Party_Of
6/17/2022	Saat 11:00.	Öğleden sonra 2:00.	Allure	Burton Düğün	60
6/17/2022	Saat 11:00.	Öğleden sonra 2:00.	Bonanza	Adams Ofis	12
6/17/2022	Öğleden sonra 3:00.	17:30'da.	Allure	Smith Ailesi	15
6/17/2022	15:30.	17:30'da.	Bonanza	Adams Ofis	12
6/18/2022	13:00.	Öğleden sonra 3:00.	Bonanza	İzciler	33
6/18/2022	Saat 11:00.	Öğleden sonra 2:00.	Allure	March of Dimes	25
6/18/2022	Saat 11:00.	12:30.	Bonanza	Smith Ailesi	12

Tablo 5.4'te gösterilen veriler göz önüne alındığında, EVENT varlığını aşağıdaki gibi modelleyebilirsiniz: EVENT (DATE, TIME\_START, TIME\_END, ROOM, EVENT\_NAME, PARTY\_OF)

Hangi birincil anahtarı önerirsiniz? Bu durumda, modelde birincil anahtar olarak kullanılabilecek basit bir doğal anahtar yoktur. Önceki bölümlerde öğrendiğiniz birincil anahtar kavramlarına dayanarak, bu seçeneklerden birini önerebilirsiniz:

**(DATE, TIME\_START, ROOM)** veya **(DATE, TIME\_END, ROOM)**

EVENT varlığı için bileşik birincil anahtarı **(DATE, TIME\_START, ROOM)** seçtiğinizi varsayın. Ardından, bir ETKİNLİĞİN birçok KAYNAK (tablolar, projektörler, bilgisayarlar ve stantlar kullanabileceğini ve aynı KAYNAĞIN birçok ETKİNLİK için kullanılabileceğini belirlersiniz. KAYNAK varlığı aşağıdaki niteliklerle temsil edilecektir:

KAYNAK (**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 bileşik birincil anahtarla aşağıdaki gibi temsil edilecektir:

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

Artık uzun, dört öznitelikli bir bileşik birincil anahtarınız var. EVNTRSC varlığının birincil anahtarı varlığa bağlı başka bir varlık tarafından devralınırsa ne olur? Bu noktada, bileşik birincil anahtarın veritabanı uygulamasını ve program kodlamasını gereksiz yere karmaşık hale getirebileceğini görebilirsiniz.

Bir veri modelleyicisi olarak, Tablo 5.3'teki birincil anahtar yönergeleri göz önüne alındığında, muhtemelen EVENT varlığının seçilen birincil anahtarının iyi sonuç vermeyebileceğini fark etmişsinizdir. Bu durumda, EVENT varlığının seçilen birincil anahtarı gömülü semantik bilgiler içerir ve tarih, saat ve metin veri sütunlarının bir kombinasyonundan oluşur. Buna ek olarak, seçilen birincil anahtar varlığa bağlı varlıklar için uzun birincil anahtarlara neden olacaktır. Tercih edilen alternatif, sayısal, tek öznitelikli bir vekil birincil anahtar kullanmaktır.

Vekil birincil anahtarlar günümüzün karmaşık veri ortamlarında kabul gören bir uygulamadır. Özellikle doğal anahtar olmadığında, seçilen aday anahtarın gömülü anlamsal içerikleri olduğunda veya seçilen aday anahtar çok uzun ya da hantal olduğunda. Bununla birlikte, bir değiş tokuş söz konusudur: bir vekil anahtar kullanırsanız, söz konusu varlığın aday anahtarının "benzersiz izin" ve "boş değil" kısıtlamalarını kullanarak düzgün bir şekilde çalışmasını sağlamanız gerekir. Yukarıdaki örnekte, (DATE, TIME\_START, ROOM, RSC\_ID) üzerinde benzersiz bir izin oluşturacaksınız.

## Not

Bu örnek, varlık bütünlüğünün korunduğu ancak iş kurallarının anlamsal doğruluğunun korunmadığı bir durumu göstermektedir. Örneğin, üst üste gelen ve birincil anahtarları tamamen uyumlu olan iki olayınız olabilir. Bu tür bir iş kuralına (iki olay üst üste gelemez-aynı odada aynı anda gerçekleşemez) uyulmasını sağlamanın tek yolu uygulama programlama kodu olacaktır.

## 5-4 Tasarım Örnekleri: Esnek Veritabanı Tasarımını Öğrenme

Veri modelleme ve veritabanı tasarımı, deneyim yoluyla edinilen beceriler gerektirir. Tecrübe ise, öğrenilen kavramların belirli ve farklı tasarım problemlerine uygulanarak düzenli ve sık tekrarlanan pratiklerle kazanılır. Bu bölümde esnek tasarımların, birincil anahtarların doğru tanımlanmasının ve yabancı anahtarların yerleştirilmesinin önemini vurgulayan dört özel tasarım vakası sunulmaktadır.

## Not

Bu kitap boyunca çeşitli modelleme kavramları açıklanırken, ilişkisel odaklanılmaktadır. Ayrıca, veritabanı tasarımının pratik doğasına odaklanıldığı için, tüm tasarım konuları uygulama hedefi göz önünde bulundurularak ele alınmaktadır. Bu nedenle, tasarım ve uygulama arasında keskin bir sınır çizgisi yoktur.

Tasarımın saf kavramsal aşamasında, yabancı anahtarlar ERD'nin bir parçası değildir. ERD yalnızca varlıkları ve ilişkileri gösterir. Varlık örnekleri, birincil anahtarlar haline gelebilecek tanımlayıcılar ile ayırt edilir. Tasarım sırasında, modelleyici varlıkları ve ilişkileri anlamaya ve tanımlamaya çalışır. Yabancı anahtarlar, ERD'de tasarlanan ilişkinin ilişkisel bir modelde uygulanmasını sağlayan mekanizmadır.

### 5-4a Tasarım Örneği 1: 1:1 İlişkilerin Uygulanması

Yabancı anahtarlar, ilişkisel modelde ilişkileri düzgün bir şekilde uygulamak için birincil anahtarlarla birlikte çalışır. Temel kural çok basittir: "bir" tarafın (ana varlık) birincil anahtarını "çok" tarafa (bağımlı varlık) yabancı anahtar olarak yerleştirin. Ancak, 1:1 ilişki ile çalışırken yabancı anahtarı nereye yerleştirirsiniz? Örneğin, "bir ÇALIŞAN bir DEPARTMANIN yöneticisidir ve bir DEPARTMAN bir ÇALIŞAN tarafından yönetilir" iş kuralına dayalı olarak ÇALIŞAN ile DEPARTMAN arasında 1:1 ilişki olduğunu varsayalım. Bu durumda, yabancı anahtarı seçmek ve yerleştirmek için iki seçenek vardır:

1. *Her iki varlığa da bir yabancı anahtar yerleştirin.* Bu seçenek, Bölüm 4'te öğrendiğiniz temel kuraldan türetilmiştir. EMP\_NUM ögesini DEPARTMENT ögesine yabancı anahtar olarak yerleştirin ve DEPT\_ID ögesini EMPLOYEE ögesine yabancı anahtar olarak yerleştirin. Ancak, bu çözüm önerilmez çünkü işi tekrarlar ve mevcut diğer ilişkilerle çakışabilir (DEPARTMENT ve EMPLOYEE'nin de 1:M ilişkisine katıldığını - bir departman birçok çalışanı istihdam eder).
2. *Varlıklardan birine bir yabancı anahtar yerleştirin.* Bu durumda, iki varlıktan birinin birincil anahtarı diğer varlıkta yabancı anahtar olarak görünür. Bu tercih edilen çözümdür, ancak bir soru kalır: yabancı anahtar olarak *hangi* birincil anahtar kullanılmalıdır? Cevap, ERD'deki ilişki özelliklerine dayalı olarak 1:1 ilişkide yabancı anahtarın seçilme gerekçesini gösteren Tablo 5.5'te bulunmaktadır.

**Tablo 5.5** 1:1 İlişkide Yabancı Anahtar Seçimi

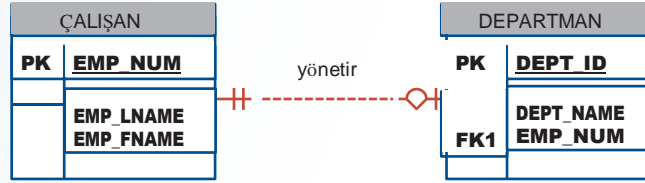
Dava	ER İlişki Kısıtlamaları	Eylem
I	Bir taraf zorunlu, diğer taraf ise isteğe bağlıdır.	Zorunlu taraftaki varlığın PK'sını isteğe bağlı taraftaki varlığa bir FK olarak yerleştirin ve FK'yı zorunlu hale getirin.
II	Her iki taraf da isteğe bağlıdır.	En az boşluğa neden olan FK'yı seçin veya FK'yı (ilişki) rolünün oynandığı yerleştirin.
III	Her iki taraf da zorunludur.	Vaka II'ye bakın veya iki varlığın tek bir varlıkta bir arada bulunmadığından emin için modelinizi gözden geçirmeyi düşünün.

Şekil 5.8'de "EMPLOYEE manages DEPARTMENT" ilişkisi gösterilmektedir. Bu durumda, EMPLOYEE'nin DEPARTMENT için zorunlu olduğuna dikkat edin. Bu nedenle, EMP\_NUM DEPARTMENT içinde yabancı anahtar olarak yerleştirilmiştir. Alternatif olarak, "yönetici" rolünün DEPARTMENT içindeki EMPLOYEE tarafından oynandığını da iddia edebilirsiniz.

## Şekil 5.8 DEPARTMAN ve ÇALIŞAN Arasındaki 1:1 İlişki

Bire Bir (1:1) İlişki:

Bir ÇALIŞAN sıfır veya bir DEPARTMAN yönetir; her DEPARTMAN bir ÇALIŞAN tarafından yönetilir.



Bir tasarımcı olarak, 1:1 ilişkilerinin gerçek dünyada var olduğunu kabul etmelisiniz; bu nedenle, veri modelinde desteklenmelidirler. Aslında, 1:1 ilişkisi iki varlık kümesinin aynı tabloya yerleştirilmemesini sağlamak için kullanılır. Başka bir deyişle, EMPLOYEE ve DEPARTMENT açıkça ayrı ve benzersiz varlık tipleridir ve tek bir varlıkta bir araya gelemezler. Bunları tek bir varlıkta gruplandırsaydınız, bu varlığa ne ad verirdiniz?

### 5-4b Tasarım Örneği 2: Zamanla Değişen Verilerin Geçmişinin Korunması

Şirket yöneticileri genellikle iyi karar vermenin veri tabanlarında depolanan veriler aracılığıyla üretilen bilgilere dayandığını farkındadır. Bu veriler hem güncel hem de geçmiş olayları yansıtır. Şirket yöneticileri veritabanlarında saklanan verileri "Şirketin mevcut kârı önceki yıllara kıyasla nasıl?" ve "XYZ ürününün satış eğilimleri nedir?" gibi soruları yanıtlamak için kullanır. Başka bir deyişle, veritabanlarında depolanan veriler yalnızca güncel verileri değil, aynı zamanda geçmiş verileri de yansıtır.

Normalde veri değişiklikleri, önceki değer dikkate alınmaksızın mevcut öznitelik değerinin yeni değerle değiştirilmesiyle yönetilir. Ancak, bazı durumlarda belirli bir öznitelige ait değerlerin geçmişi korunmalıdır. Veri modelleme açısından, **zaman açısından değişken veriler**, değerleri zaman içinde değişen ve veri değişikliklerinin geçmişini tutmanız *gerek* veriler anlamına gelir. Bir veritabanındaki tüm verilerin zaman içinde değişime tabi olduğunu ve bu nedenle zaman değişkenli olduğunu iddia edebilirsiniz. Ancak, doğum tarihiniz veya Sosyal Güvenlik numaranız gibi bazı öznitelik değerleri zaman değişkenli değildir. Öte yandan, öğrenci not ortalamanız veya banka hesap bakiyeniz gibi öznitelikler zaman içinde değişime tabidir. Bazen veri değişiklikleri, ürün fiyatı değişikliği gibi dış kaynaklı ve olay odaklıdır. Diğer durumlarda, değişiklikler günlük hisse senedi kotasyonu "açılış" ve "kapanış" değerleri gibi iyi tanımlanmış programlara dayanır.

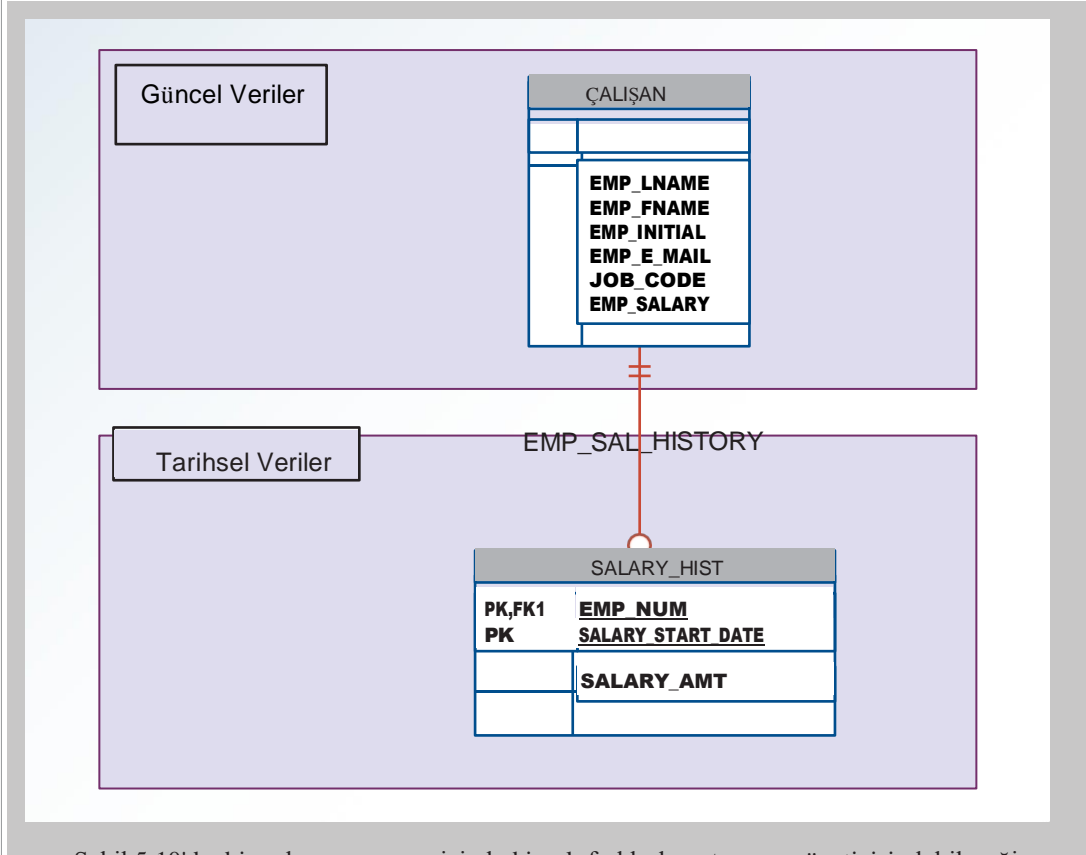
Zamana göre değişen verilerin depolanması veri modelinde değişiklik yapılmasını gerektirir; değişikliğin türü verinin yapısına bağlıdır. Bazı zaman değişkenli veriler, varlığınızda çok değerli bir öznitelige sahip olmaya eşdeğerdir. Bu tür zaman değişkenli verileri modellemek için, orijinal varlıkla 1:M ilişkisi içinde yeni bir varlık oluşturmamız gerekir. Bu yeni varlık yeni değeri, değişikliğin tarihini ve modellenen olayla ilgili diğer öznitelikleri içerecektir. Örneğin, her bir çalışan için maaş geçmişlerini takip etmek istiyorsanız, EMP\_SALARY özniteliği Şekil 5.9'da gösterildiği gibi çok değerli hale gelir. Bu durumda, her çalışan için SALARY\_HIST varlığında maaş tutarını ve yeni maaşın yürürlüğe girdiği tarihi saklayan bir veya daha fazla kayıt olacaktır.

Diğer zaman değişkenli veriler 1:M ilişkisini M:N ilişkisine dönüştürebilir. Çalışan verilerine ek olarak, veri modelinizin kuruluştaki farklı departmanlar ve her departmanı hangi çalışanın yönettiği hakkında veriler içerdiğini varsayın. Her departmanın yalnızca bir çalışan tarafından yönetildiğini ve her çalışanın en fazla bir departmanı yönetebileceğini varsayarsak, ÇALIŞAN ile DEPARTMENT arasında 1:1 ilişkisi olacaktır. Bu ilişki her departmanın mevcut yöneticisini kaydedecektir. Ancak, mevcut yöneticinin yanı sıra tüm departman yöneticilerinin geçmişini de takip etmek istiyorsanız, Şekil 5.10'da gösterilen modeli oluşturabilirsiniz.

**zaman değişkenli veriler** Değerleri zamanın bir fonksiyonu olan veriler. İçin Örneğin, zamana göre değişen veriler, bir şirketin tüm geçmiş idari atamalar takip edilmektedir.



## Şekil 5.9 Maaş Geçmişinin Korunması

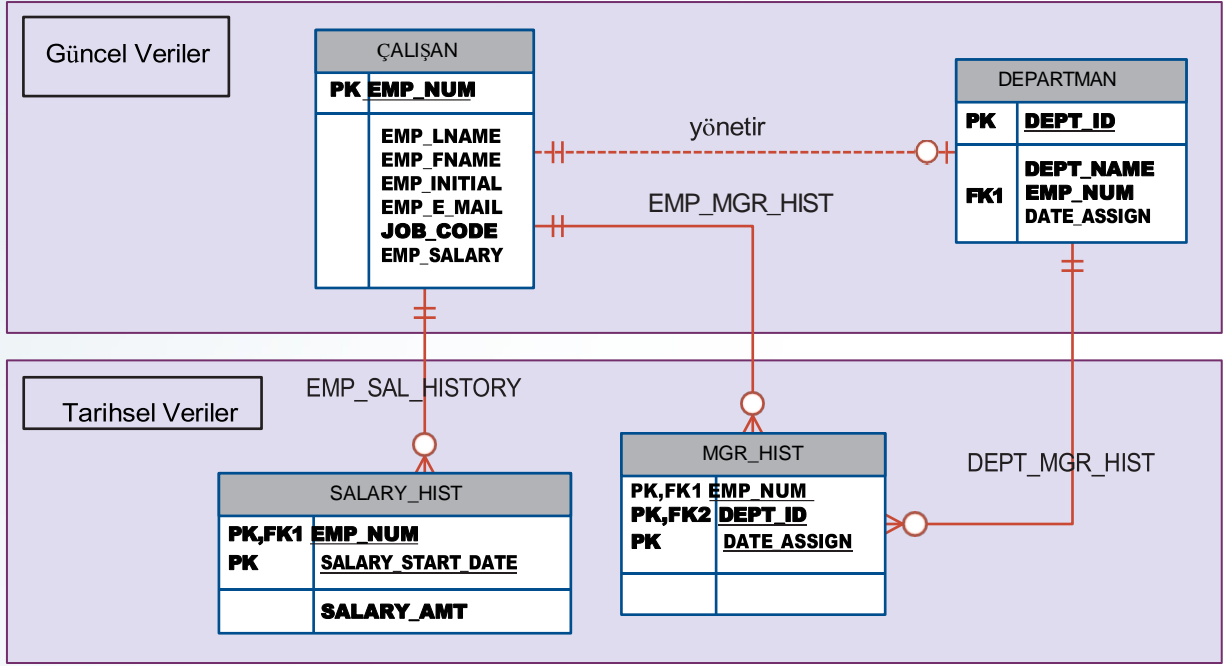


Şekil 5.10'da, bir çalışanın zaman içinde birçok farklı departmanın yöneticisi olabileceği ve bir departmanın birçok farklı çalışan yöneticisi olabileceği gerçeğini yansıtmak için MGR\_HIST varlığının EMPLOYEE ile 1:M ilişkisine ve DEPARTMENT ile 1:M ilişkisine sahip olduğuna dikkat edin. Zamana göre değişen verileri kaydettiğiniz için, çalışanın (EMP\_NUM) departman yöneticisi olduğu tarihi sağlamak üzere MGR\_HIST varlığında DATE\_ASSIGN öznitelikliğini saklamanız gerekir. MGR\_HIST'in birincil anahtarı, aynı çalışanın farklı tarihlerde aynı departmanın yöneticisi olmasına izin verir. Ortamınızda böyle bir senaryo söz konusu değilse - örneğin, bir çalışan yalnızca bir kez departman müdürü olduysa - DATE\_ASSIGN ögesini MGR\_HIST varlığında birincil olmayan bir öznitelik haline getirebilirsiniz.

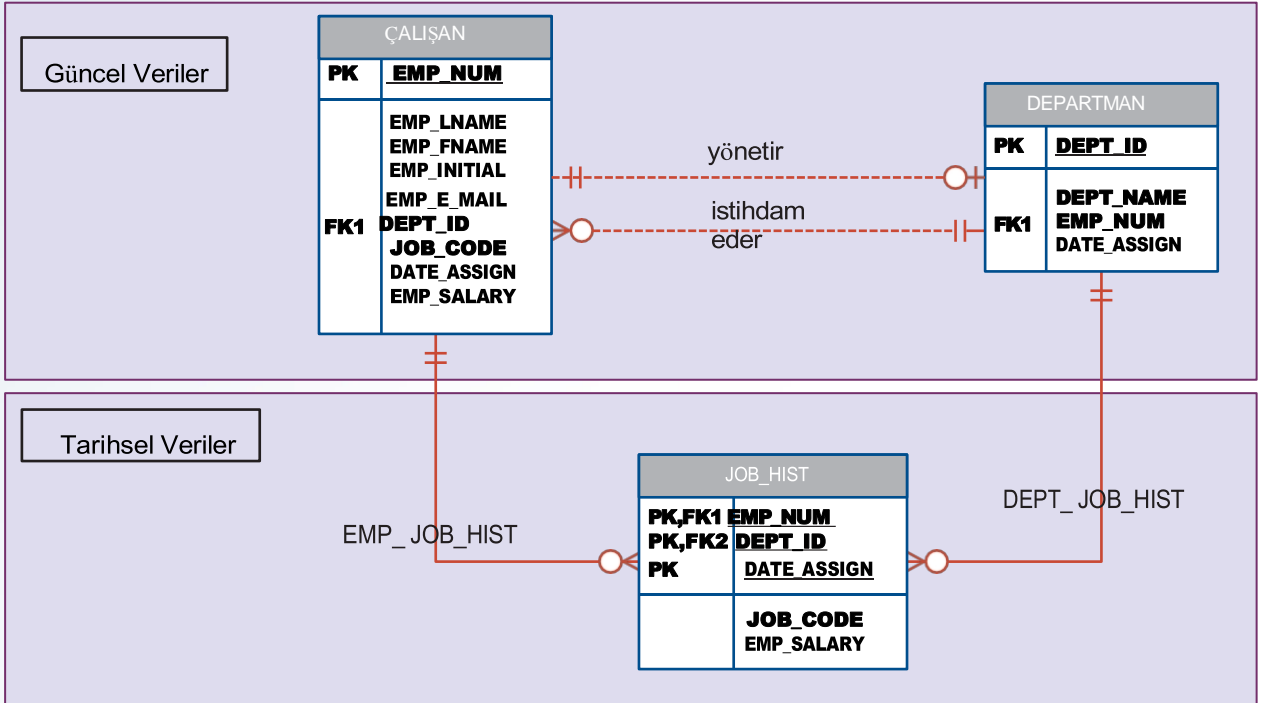
Şekil 5.10'da "manages" ilişkisinin teoride isteğe bağlı ve pratikte gereksiz olduğuna dikkat edin. Herhangi bir zamanda, belirli bir departman için MGR\_HIST'ten en son DATE\_ASSIGN tarihini alarak bir departmanın yöneticisini belirleyebilirsiniz. Öte yandan, Şekil 5.10'daki ERD, güncel veriler ile geçmiş verileri birbirinden ayırır. *Mevcut* yönetici ilişkisi, EMPLOYEE ve DEPARTMENT arasındaki "manages" ilişkisi tarafından uygulanmaktadır. Ayrıca, geçmiş veriler EMP\_MGR\_HIST ve DEPT\_MGR\_HIST aracılığıyla yönetilir. Bu modelin dezavantajı, bir departmana her yeni yönetici atandığında iki veri değişikliği yapılmasıdır: DEPARTMENT varlığında bir güncelleme ve MGR\_HIST varlığında bir ekleme.

Şekil 5.10'da önerilen modelin esnekliği, 1:M "bir departman birçok çalışanı istihdam eder" ilişkisini eklediğinizde daha belirgin hale gelir. Bu durumda, "1" tarafının PK'sı (DEPT\_ID) "çok" tarafında (EMPLOYEE) yabancı anahtar olarak görünür. Şimdi, şirket çalışanlarının her biri için iş geçmişini takip etmek istediğinizi varsayalım; muhtemelen departmanı, iş kodunu, atanan tarihi ve maaşı saklamak istersiniz. Bu görevi yerine getirmek için, Şekil 5.10'daki modeli bir JOB\_HIST varlığı ekleyerek değiştirebilirsiniz. Şekil 5.11, çalışanın geçmişini tutmak için yeni JOB\_HIST varlığının kullanımını göstermektedir.

Şekil 5.10 Yönetici Geçmişinin Korunması



Şekil 5.11 İş Geçmişinin Korunması



Yine, "yönetir" ve "istihdam eder" ilişkilerinin teorik olarak isteğe bağlı ve pratikte gereksiz olduğunu vurgulamakta fayda var. İş geçmişine bakarak ve her için yalnızca en güncel veri satırını seçerek her çalışanın nerede çalıştığını her zaman öğrenebilirsiniz. Ancak, Bölüm 7, Yapısal Sorgu Diline (SQL) Giriş'te keşfedeceğiniz gibi ve

#### tasarım tuzağı

Bir ilişki yanlış veya eksik tanımlandığında ve dolayısıyla gerçek dünya ile tutarlı olmayan bir şekilde temsil edildiğinde ortaya çıkan bir sorun. Bu

En yaygın tasarım tuzağı *fan tuzağı* olarak bilinir.

#### fan kapanı

Bir varlık diğer varlıklarla iki 1:M ilişkisi olduğunda ortaya çıkan ve böylece diğer varlıklar arasında modelde ifade edilmeyen bir ilişki üreten bir tasarım tuzağı.

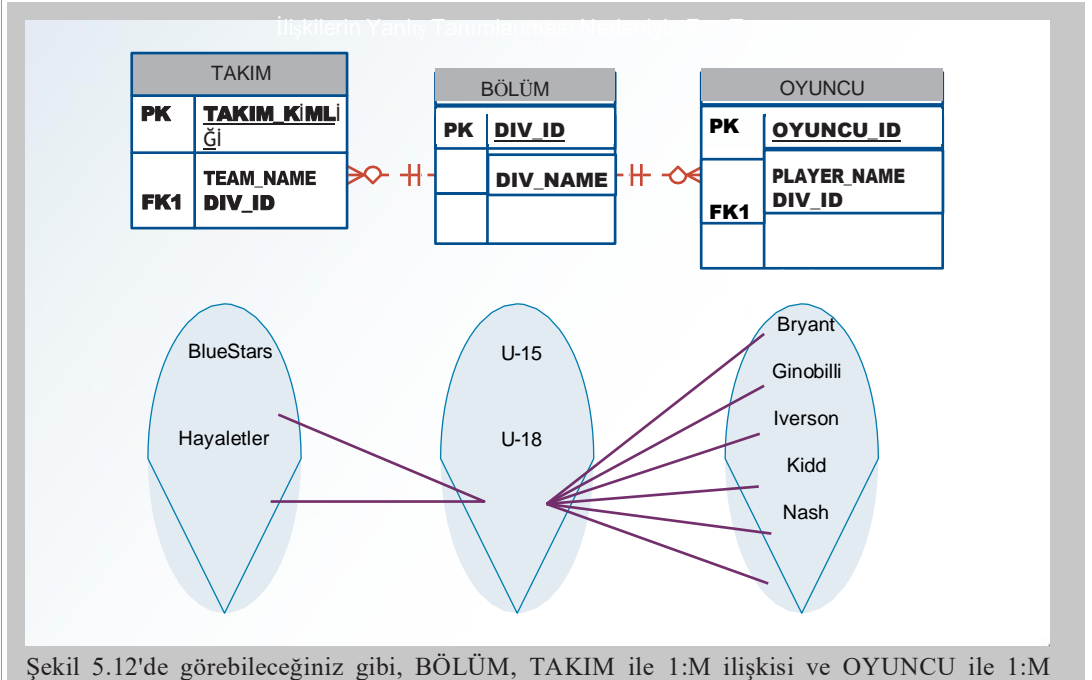
Bölüm 8, Gelişmiş SQL'de, her bir çalışanın nerede çalıştığını bulmak önemsiz bir görev değildir. Bu nedenle, Şekil 5.11'de gösterilen model, mevcut verileri geçmiş verilerden ayırmak için kuşkusuz gereksiz ancak son derece yararlı "manages" ve "employs" ilişkilerini içerir.

### 5-4c Tasarım Örneği 3: Fan Kapanları

Bir veri modeli oluşturmak, varlıklar arasındaki veri ilişkilerinin doğru bir şekilde tanımlanmasını gerektirir. Ancak, iletişimsizlik veya iş kuralları ya da süreçlerinin tam olarak anlaşılabilmesi nedeniyle, varlıklar arasındaki ilişkilerin yanlış tanımlanması nadir değildir. Bu koşullar altında ERD bir tasarım tuzağı içerebilir. **Tasarım tuzağı**, bir ilişki yanlış veya eksik tanımlandığında ve bu nedenle gerçek dünyayla uyumlu olmayan bir şekilde temsil edildiğinde ortaya çıkar. En yaygın tasarım tuzağı *fan tuzağı* olarak bilinir.

**Taraftar tuzağı**, bir varlığın diğer varlıklarla iki 1:M ilişkisi içinde olması ve böylece diğer varlıklar arasında modelde ifade edilmeyen bir ilişki üretilmesi durumunda ortaya çıkar. Örneğin, JCB basketbol liginin birçok bölümü olduğunu varsayın. Her ligde çok sayıda oyuncu ve her ligde çok sayıda takım vardır. Bu "eksik" iş kuralları göz önüne alındığında, Şekil 5.12'deki gibi görünen bir ERD oluşturabilirsiniz.

Şekil 5.12 Fan Kapanı Sorunu ile Yanlış ERD

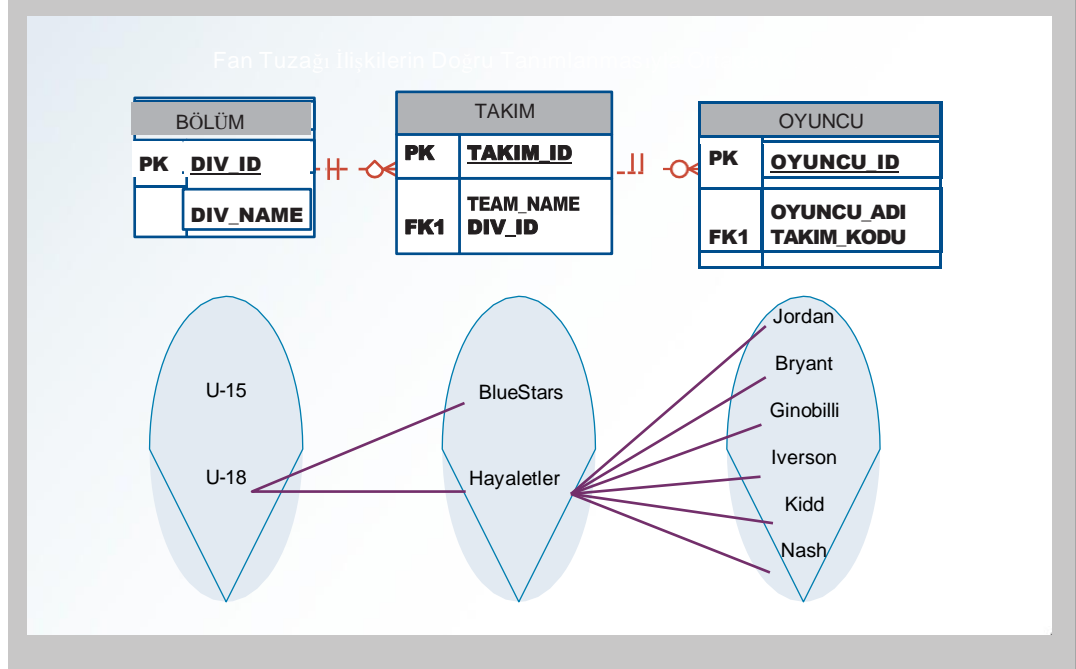


Şekil 5.12'de görebileceğiniz gibi, BÖLÜM, TAKIM ile 1:M ilişkisi ve OYUNCU ile 1:M ilişkisi içindedir. Bu gösterim anlamsal olarak doğru olsa da, ilişkiler düzgün bir şekilde tanımlanmamıştır. Örneğin, hangi oyuncunun hangi takıma ait olduğunu belirlemenin bir yolu yoktur. Şekil 5.12 ayrıca ERD için örnek bir ilişki gösterimini de göstermektedir. DIVISION örnekleri için ilişki çizgilerinin TEAM ve PLAYER varlık örneklerine doğru yayıldığına dikkat edin - bu nedenle "fan trap" etiketi.

Şekil 5.13, fan tuzağı ortadan kaldırıldıktan sonra doğru ERD'yi göstermektedir. Bu durumda, DIVISION'ın TEAM ile 1:M ilişkisi içinde olduğuna dikkat edin. Buna karşılık, TEAM de PLAYER ile 1:M ilişkisi içindedir. Şekil 5.13 ayrıca fan tuzağı ortadan kaldırıldıktan sonraki örnek ilişki gösterimini de göstermektedir.

Şekil 5.13'teki tasarım göz önüne alındığında, hangi oyuncuların hangi takımda oynadığını görmenin ne kadar kolay olduğuna dikkat edin. Ancak, hangi oyuncuların hangi ligde oynadığını bulmak için, önce her ligde hangi takımların olduğunu görmemiz ve ardından her takımda hangi oyuncuların oynadığını bulmanız gerekir. Başka bir deyişle, TEAM varlığı aracılığıyla DIVISION ve PLAYER arasında geçişli bir ilişki vardır.

Şekil 5.13 Fan Kapanı Söküldükten Sonra Düzeltilmiş ERD



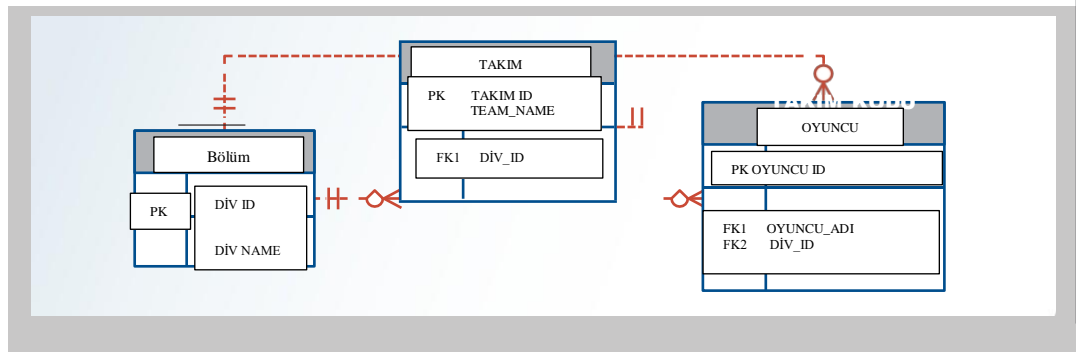
#### 5-4d Tasarım Örneği 4: Gereksiz İlişkiler

Bilgisayar ortamlarında fazlalık genellikle iyi bir şey olsa da (örneğin birden fazla yerde birden fazla yedekleme), veritabanı ortamında fazlalık nadiren iyi bir şeydir. (Bölüm 3, İlişkisel Veritabanı Modeli'nde öğrendiğiniz gibi, fazlalıklar bir veritabanında veri anomalilerine neden olabilir). Gereksiz ilişkiler, ilgili varlıklar arasında birden fazla ilişki yolu olduğunda ortaya çıkar. Gereksiz ilişkilerle ilgili temel endişe, bunların model genelinde tutarlı kalmasıdır. Bununla birlikte, bazı tasarımların gereksiz ilişkileri tasarımı basitleştirmenin bir yolu olarak kullandığını belirtmek önemlidir.

Gereksiz ilişkilerin bir örneği ilk olarak Şekil 5.10'da zaman değişkenli verilerin geçmişinin tutulması tartışması sırasında ortaya konmuştur. Ancak, gereksiz "yönetir" ve "kullanır" ilişkilerinin kullanımı, bu tür ilişkilerin geçmiş verilerden ziyade mevcut verilerle ilgilenmesi gerçeğiyle gereğeldenir. Gereksiz ilişkilerin daha spesifik bir başka örneği Şekil 5.14'te gösterilmektedir.

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, modelde herhangi bir bilgi üretme kabiliyetini kaybetmeden ilişki güvenli bir şekilde silinebilir. Oyuncunun bölümünü TEAM varlığı aracılığıyla belirleyebileceğinizi unutmayın.

#### Gereksiz Bir İlişki



- Genişletilmiş varlık ilişkisi (EER) modeli, varlık üst tipleri, alt tipleri ve kümeleri aracılığıyla ER modeline anlamlılık katar. Bir varlık üst tipi, bir veya daha fazla varlık alt tipiyle ilişkili olan genel bir varlık tipidir.
- Bir uzmanlaşma hiyerarşisi, varlık üst tipleri ile varlık alt tipleri arasındaki düzenlemeyi ve ilişkileri tasvir eder. Kalıtım, bir varlık alt tipinin üst tipin niteliklerini ve ilişkilerini miras aldığı anlamına gelir. Alt tipler ayrı veya örtüşen olabilir. Bir alt tip ayrıcı, üst tip oluşumunun hangi varlık alt tipiyle ilişkili olduğunu belirlemek için kullanılır. Alt tipler kısmi veya tam bütünlük sergileyebilir. Varlık üst tipleri ve alt tiplerinin uzmanlaşma hiyerarşisini geliştirmek için temelde iki yaklaşım vardır: uzmanlaşma ve genelleştirme.
- Varlık kümesi, ERD'de birden fazla varlığı ve ilişkiyi temsil etmek için kullanılan "sanal" bir varlık türüdür. Bir varlık kümesi, birbiriyle ilişkili birden fazla varlık ve ilişkinin tek bir soyut varlık nesnesinde birleştirilmesiyle oluşturulur.
- Doğal anahtarlar gerçek dünyada var olan tanımlayıcılardır. Doğal anahtarlar bazen iyi birincil anahtarlar olabilir, ancak her zaman değil. Birincil anahtarlar benzersiz değerlere sahip olmalı, akıllı olmamalı, zaman içinde değişmemeli ve tercihen sayısal olmalı ve tek bir öznelikten .
- Bileşik anahtarlar M:N ilişkilerini ve zayıf (güçlü tanımlayıcı) varlıkları temsil etmek için kullanışlıdır.
- Vekil birincil anahtarlar, uygun bir birincil anahtar oluşturan doğal bir anahtar olmadığında, birincil anahtar birden fazla veri türüne sahip bileşik bir birincil anahtar olduğunda veya birincil anahtar kullanılamayacak kadar uzun olduğunda kullanışlıdır.
- 1:1 ilişkide, zorunlu varlığın PK'sını isteğe bağlı varlıkta yabancı anahtar olarak, en az boşluğa neden olan varlıkta bir FK olarak veya rolün oynandığı yerde bir FK olarak yerleştirin.
- Zamana göre değişen veriler, değerleri zaman içinde değişen ve veri değişikliklerinin geçmişini tutmanızı gerektiren verileri ifade eder. Zamana göre değişen verilerin geçmişini tutmak için, yeni değeri, değişiklik tarihini ve zamanla ilgili diğer verileri içeren bir varlık oluşturmamız gerekir. Bu varlık, geçmişi tutulacak varlıkla 1:M ilişkisini korur.
- Fan tuzağı, bir varlığın diğer varlıklarla iki 1:M ilişkisi olduğunda ve diğer varlıklar arasında modelde ifade edilmeyen bir olduğunda ortaya çıkar. Gereksiz ilişkiler, ilgili varlıklar arasında birden fazla ilişki yolu olduğunda ortaya çıkar. Gereksiz ilişkilerle ilgili temel endişe, bunların model genelinde tutarlı kalmasıdır.

## Anahtar Terimler

bütünlük kısıtı tasarım tuzağı  
ayrık alt tip  
EER diyagramı  
(EERD) varlık kümesi  
varlık alt türü varlık  
üst türü

genişletilmiş varlık ilişki modeli  
(EERM)  
fan tuzağı genelleme  
kalıtım  
doğal anahtar (doğal tanımlayıcı)  
örtüşmeyen alt tip örtüşen alt tip

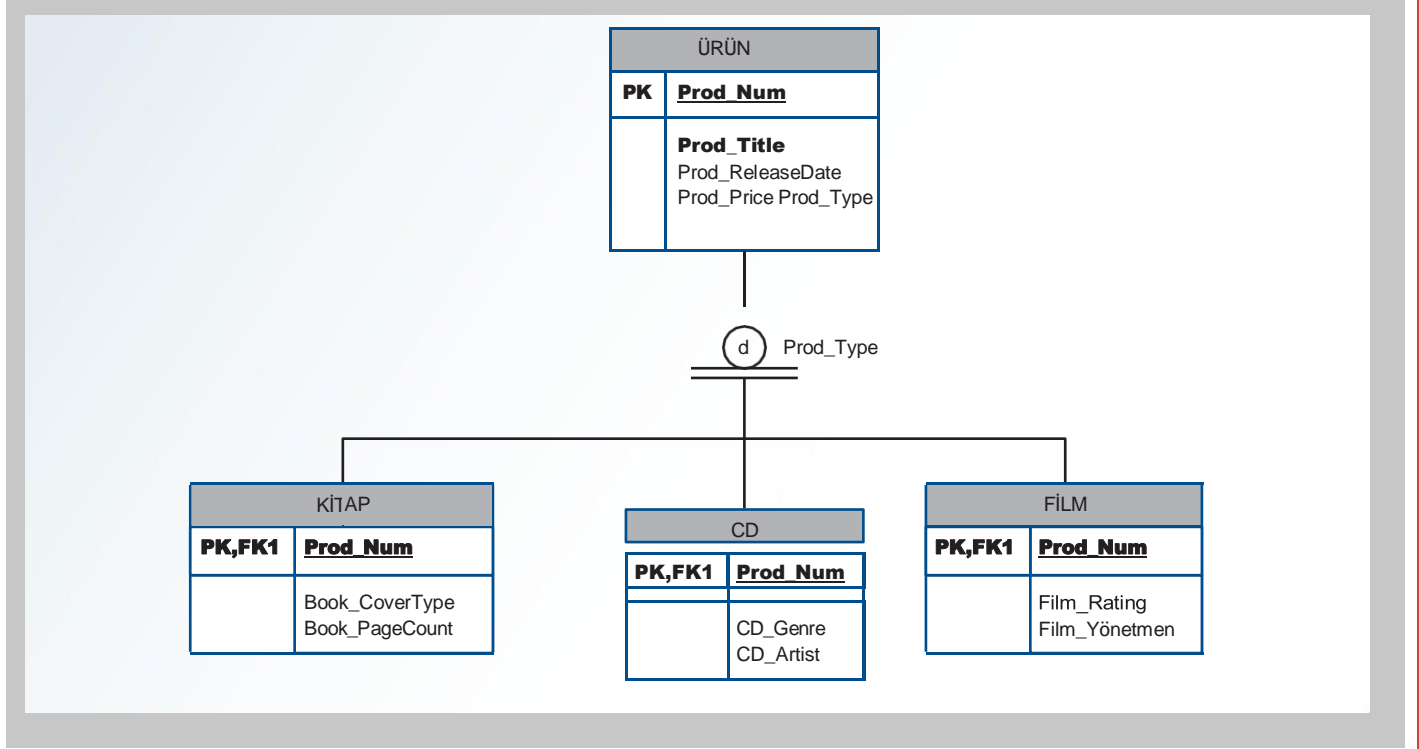
kısmi tamlık özelleştirme  
özelleştirme hiyerarşisi alt  
tür ayrıcı vekil anahtar  
zaman değişkenli veri  
toplam tamlık

## İnceleme Soruları

1. Varlık süper tipi nedir ve neden kullanılır?
2. Bir varlık alt türünde ne tür veriler depolarsınız?
3. Uzmanlaşma hiyerarşisi nedir?
4. Alt tip ayrıcı nedir? Kullanımına bir örnek veriniz.
5. Örtüşen alt tip nedir? Bir örnek veriniz.
6. Ayrık alt tip nedir? Bir örnek veriniz.
7. Kısmi tamlık ile tam tamlık arasındaki fark nedir?

Soru 8-10 için Şekil Q5.8'e bakınız.

**Şekil S5.8 ÜRÜN Veri Modeli**



8. Bir filmin tüm özelliklerini listeleyin.
9. Veri modeline göre, PRODUCT tablosundaki her varlık örneğinin CD tablosundaki bir varlık örneği ile ilişkilendirilmesi gerekli midir? Neden ya da neden değil?
10. Bir kitabın ÜRÜN tablosunda görünmeden KİTAP tablosunda görünmesi mümkün müdür? Neden ya da neden olmasın?
11. Varlık kümesi nedir ve kullanımından ne gibi avantajlar elde edilir?
12. Hangi temel özellikler arzu edilir olarak kabul edilir? Her bir özelliğin neden arzu edilir olarak değerlendirildiğini açıklayın.
13. Bileşik birincil anahtarlar hangi koşullar altında uygundur?
14. Vekil birincil anahtar nedir ve ne zaman kullanırsınız?
15. 1:1 ilişkisini uygularken, bir taraf zorunlu ve bir taraf isteğe bağlıysa yabancı anahtarı nereye yerleştirmelisiniz? Yabancı anahtar zorunlu mu yoksa isteğe bağlı mı olmalıdır?
16. Zamanla değişen veri nedir ve veritabanı tasarımı açısından bu tür verilerle nasıl başa çıkarsınız?
17. En yaygın tasarım tuzağı nedir ve nasıl ortaya çıkar?

## Problemler

1. Aşağıdaki iş senaryosunu göz önünde bulundurarak, uygunsa bir uzmanlaşma hiyerarşisi kullanarak bir Crow's Foot ERD oluşturun. Two-Bit Drilling Company, çalışanlar ve onların sigortalı bakmakla yükümlü oldukları kişiler hakkındaki bilgileri tutar. Her çalışanın bir çalışan numarası, adı, işe alınma tarihi ve unvanı vardır. Bir çalışan müfettiş ise, sertifika tarihi ve sertifika yenileme tarihi de sisteme kaydedilmelidir. Tüm çalışanlar için Sosyal Güvenlik numarası ve bakmakla yükümlü olunan kişilerin isimleri saklanmalıdır. Tüm bakmakla yükümlü olunan kişiler bir ve yalnızca bir çalışanla ilişkilendirilmelidir. Bazı çalışanların bakmakla yükümlü olduğu kişi olmazken, bazılarının çok sayıda bakmakla yükümlü olduğu kişi olacaktır.
2. Aşağıdaki iş senaryosunu göz önüne alarak, uygunsa bir uzmanlaşma hiyerarşisi kullanarak bir Crow's Foot ERD oluşturun. Tiny Hospital, hastalar ve hastane odaları hakkında bilgi tutmaktadır. Sistem her hastaya bir hasta kimlik numarası atar. Ayrıca hastanın adı ve doğum tarihi de kaydedilir. Bazı hastalar en az bir yıl hastanede kalan yerleşik hastalardır.



ÇEVİRİYİ + SORU YAPAN	UMUT TÜRKER
SUNUM + DÜZENLEME +SORU YAPAN	ABDULKADİR ÖCAL