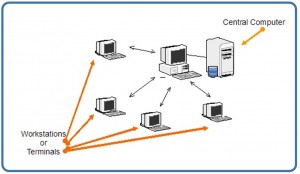
**Veritabanı Dağıtımına Dayalı Sınıflandırma**

Veritabanı sistemleri için dört ana dağıtım sistemi vardır ve bunlar da VTYS'yi sınıflandırmada kullanılabilir.

Merkezi sistemler

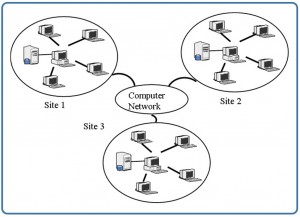
*Merkezi* bir veritabanı sistemiyle , DBMS ve veritabanı, birkaç başka sistem tarafından da kullanılan tek bir sitede saklanır. Bu, Şekil 6.1'de gösterilmiştir.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Record-300x177.jpg)Şekil 6.1. Merkezi bir veritabanı sistemine örnek.

1980'lerin başlarında, birçok Kanada kütüphanesi, manuel kart kataloglarını makine tarafından okunabilen merkezi katalog sistemlerine dönüştürmek için GEAC 8000'i kullandı. Her kitap kataloğunun süpermarket ürünlerindekine benzer bir barkod alanı vardı.

Dağıtık veritabanı sistemi

Dağıtılmış bir *veritabanı sisteminde* , gerçek veritabanı ve DBMS yazılımı, Şekil 6.2'de gösterildiği gibi, bir bilgisayar ağıyla birbirine bağlı çeşitli sitelerden dağıtılır.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/EntitySet-300x86.jpg)Şekil 6.2. Dağıtık bir veritabanı sisteminin örneği.

Homojen dağıtılmış veritabanı sistemleri

*Homojen dağıtılmış veritabanı sistemleri* birden fazla siteden aynı DBMS yazılımını kullanır. Bu çeşitli siteler arasındaki veri alışverişi kolayca gerçekleştirilebilir. Örneğin, Geac Computer Corporation gibi aynı satıcının kütüphane bilgi sistemleri, çeşitli Geac kütüphane siteleri arasında kolay veri alışverişine izin veren aynı DBMS yazılımını kullanır.

Heterojen dağıtılmış veritabanı sistemleri

*Heterojen dağıtılmış* bir veritabanı sisteminde , farklı siteler farklı DBMS yazılımları kullanabilir, ancak bu siteler arasında veri alışverişini desteklemek için ek ortak yazılımlar vardır. Örneğin, çeşitli kütüphane veritabanı sistemleri, kütüphane kayıt veri alışverişini desteklemek için aynı makine tarafından okunabilen kataloglama (MARC) biçimini kullanır.

Anahtar Terimler

**merkezi veritabanı sistemi:** DBMS ve veritabanı, birkaç başka sistem tarafından da kullanılan tek bir sitede depolanır

**dağıtılmış veritabanı sistemi** :  gerçek veritabanı ve DBMS yazılımı, bir bilgisayar ağıyla birbirine bağlı çeşitli sitelerden dağıtılır

**heterojen dağıtılmış veritabanı sistemi** :  farklı siteler farklı DBMS yazılımları kullanabilir, ancak bu siteler arasında veri alışverişini desteklemek için ek ortak yazılımlar vardır

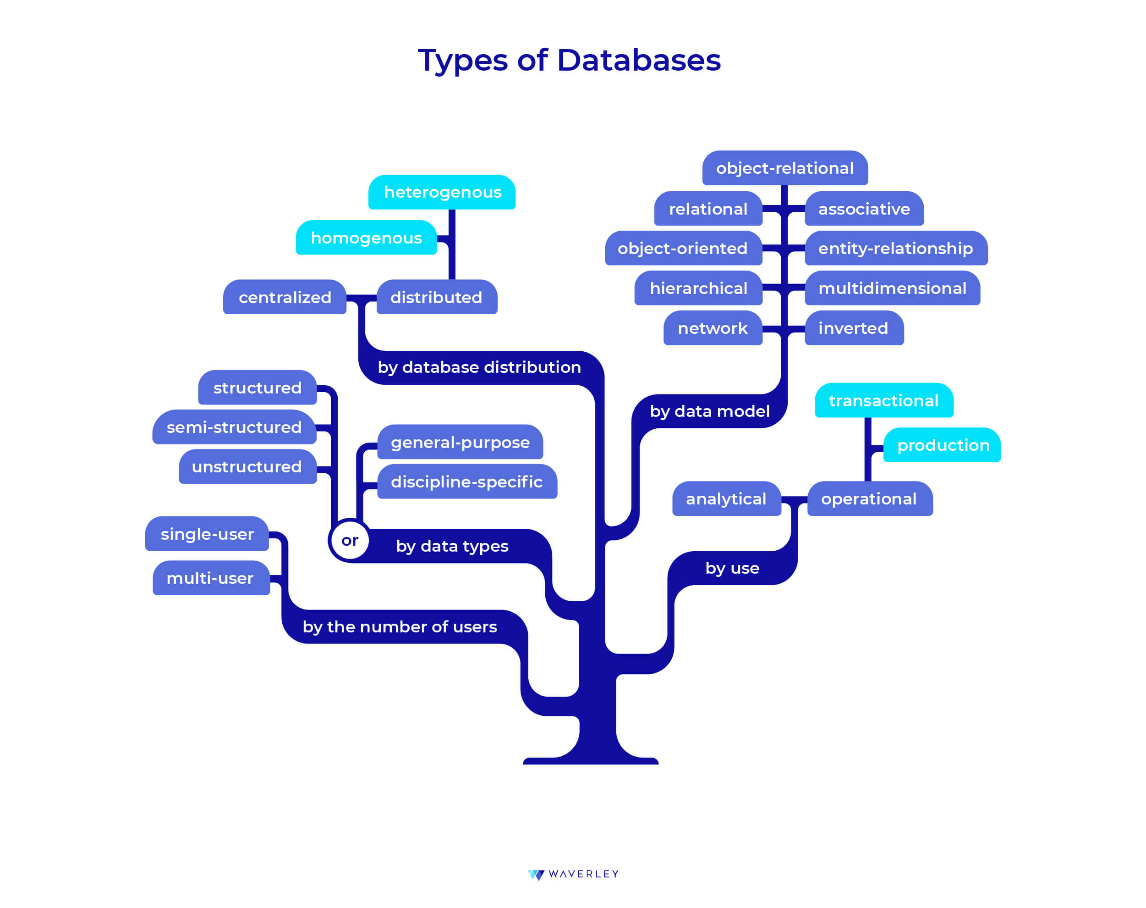
**homojen dağıtılmış veritabanı sistemleri** : birden fazla sitede aynı DBMS yazılımını kullanın

**çok kullanıcılı veritabanı sistemi** : birden fazla kullanıcıyı aynı anda destekleyen bir veritabanı yönetim sistemi

**nesne yönelimli veri modeli** : nesne yönelimli programlamada kullanılan gibi bilgilerin nesneler biçiminde temsil edildiği bir veritabanı yönetim sistemi

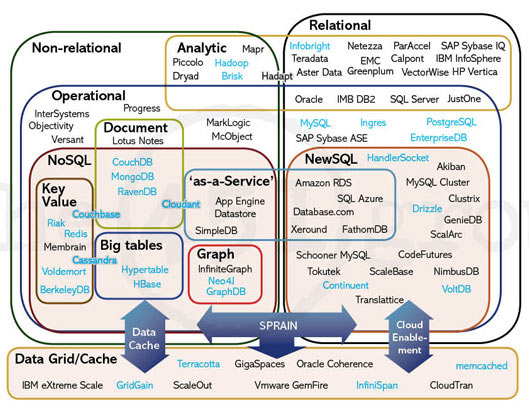
**tek kullanıcılı veritabanı sistemi** : aynı anda bir kullanıcıyı destekleyen bir veritabanı yönetim sistemi

**Geleneksel modeller** : İlişkisel modelden önceki veri modelleri



diyagram, metin, ekran görüntüsü, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu



metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Veri Modelleme**

*Veri modelleme,* veritabanı tasarımı sürecinin ilk adımıdır. Bu adım bazen kavramsal tasarım olarak da adlandırılan yüksek seviyeli ve soyut bir tasarım aşaması olarak kabul edilir . Bu aşamanın amacı şunları tanımlamaktır:

* Veritabanında bulunan veriler (örneğin, varlıklar: öğrenciler, öğretim görevlileri, dersler, konular)
* Veri öğeleri arasındaki ilişkiler (örneğin, öğrenciler öğretim görevlileri tarafından denetlenir; öğretim görevlileri ders verir)
* Verilerdeki kısıtlamalar (örneğin, öğrenci numarası tam olarak sekiz basamaklıdır; bir dersin yalnızca dört veya altı kredi birimi vardır)

İkinci adımda , veri öğeleri, ilişkiler ve kısıtlamalar, yüksek seviyeli veri modeli tarafından sağlanan kavramlar kullanılarak ifade edilir. Bu kavramlar uygulama ayrıntılarını içermediğinden, veri modelleme sürecinin sonucu veritabanı yapısının (yarı) resmi bir temsilidir. Bu sonuç anlaşılması oldukça kolaydır, bu nedenle kullanıcının tüm gereksinimlerinin karşılandığından emin olmak için referans olarak kullanılır.

Üçüncü adım veritabanı tasarımıdır . Bu adımda, iki alt adımımız olabilir: birincisi , belirli bir DBMS'nin veri modelinde bir veritabanını tanımlayan veritabanı *mantıksal tasarımı , diğeri ise dahili veritabanı depolama yapısını, dosya organizasyonunu veya dizinleme tekniklerini tanımlayan veritabanı fiziksel tasarımıdır* . Bu iki alt adım, veritabanı uygulaması ve işlemler/kullanıcı arayüzleri oluşturma adımlarıdır.

Veritabanı tasarım aşamalarında, veriler belirli bir veri modeli kullanılarak temsil edilir. *Veri modeli* , verileri, veri ilişkilerini, veri semantiğini ve veri kısıtlamalarını tanımlamak için bir kavram veya gösterim koleksiyonudur. Çoğu veri modeli ayrıca veritabanındaki verileri işlemek için bir dizi temel işlem içerir.

**Veri Soyutlamasının Dereceleri**

Bu bölümde veritabanı tasarım sürecine özgüllük açısından bakacağız. Herhangi bir tasarım yüksek bir seviyeden başlayıp giderek artan bir ayrıntı seviyesine doğru ilerlediğinden, veritabanı tasarımı da öyledir. Örneğin, bir ev inşa ederken, evin kaç yatak odası ve banyosu olacağı, tek katlı mı yoksa birden fazla katlı mı olacağı vb. ile başlarsınız. Bir sonraki adım, bir mimarın evi daha yapılandırılmış bir perspektiften tasarlamasını sağlamaktır. Bu seviye, gerçek oda boyutları, evin nasıl kablolanacağı, sıhhi tesisat armatürlerinin nereye yerleştirileceği vb. açısından daha ayrıntılı hale gelir. Son adım, evi inşa etmek için bir müteahhit tutmaktır. Bu, tasarıma yüksek bir soyutlama seviyesinden giderek artan bir ayrıntı seviyesine doğru bakmaktır.

Veritabanı tasarımı buna çok benzer. Kullanıcıların iş kurallarını tanımlamasıyla başlar; ardından veritabanı tasarımcıları ve analistler veritabanı tasarımını oluşturur; ve ardından veritabanı yöneticisi tasarımı bir DBMS kullanarak uygular.

Aşağıdaki alt bölümler, modelleri azalan soyutlama düzeyine göre özetlemektedir.

Dış modeller

* Kullanıcının veritabanına ilişkin görünümünü temsil eder
* Birden fazla farklı dış görünüm içerir
* Her kullanıcının algıladığı gerçek dünyayla yakından ilişkilidir

Kavramsal modeller

* Esnek veri yapılandırma yetenekleri sağlayın
* "Topluluk görünümü" sunun: tüm veritabanının mantıksal yapısı
* Veritabanında saklanan verileri içerir
* Aşağıdakiler dahil olmak üzere veriler arasındaki ilişkileri gösterin:
  + Kısıtlamalar
  + Anlamsal bilgiler (örneğin, iş kuralları)
  + Güvenlik ve bütünlük bilgileri
* Bir veritabanını çeşitli türdeki varlıkların (nesnelerin) bir koleksiyonu olarak düşünün
* Ana veri nesnelerinin tanımlanması ve üst düzey açıklaması için temel oluştururlar; ayrıntılardan kaçınırlar
* Kullanacağınız veritabanından bağımsız olarak veritabanından bağımsız mısınız?

Dahili modeller

Bu türün en iyi bilinen üç modeli ilişkisel veri modeli, ağ veri modeli ve hiyerarşik veri modelidir. Bu dahili modeller:

* Bir veritabanını sabit boyutlu kayıtların bir koleksiyonu olarak düşünün
* Fiziksel düzeye veya dosya yapısına daha yakındır
* Veritabanının DBMS tarafından görülen bir temsilidir.
* Tasarımcının kavramsal modelin özelliklerini ve kısıtlamalarını seçilen uygulama modelinin özellikleri ve kısıtlamalarıyla eşleştirmesini gerektirir
* Kavramsal modeldeki varlıkların ilişkisel modeldeki tablolara eşlenmesini içerir

Fiziksel modeller

* Veritabanının fiziksel temsili mi?
* En düşük soyutlama düzeyine sahip olun
* Verilerin nasıl saklandığı; bunlarla nasıl başa çıktıkları
  + Çalışma zamanı performansı
  + Depolama kullanımı ve sıkıştırma
  + Dosya organizasyonu ve erişim yöntemleri
  + Veri şifreleme
* Fiziksel düzey *işletim sistemi* ( *OS) tarafından yönetilir mi?*
* Verilerin bilgisayarın belleğinde nasıl depolandığına ilişkin ayrıntıları açıklayan kavramları sağlayın

**Veri Soyutlama Katmanı**

Resimli bir görünümde, farklı modellerin nasıl birlikte çalıştığını görebilirsiniz. Buna en üst seviyeden, harici modelden bakalım.

Harici model, son kullanıcının verilere bakış açısıdır. Genellikle bir veritabanı, birden fazla departmanın ihtiyaçlarını karşılayan bir kurumsal sistemdir. Ancak, bir departman diğer departmanların verilerini görmekle ilgilenmez (örneğin, insan kaynakları (İK) departmanı satış departmanının verilerini görmek istemez). Bu nedenle, bir kullanıcı görünümü diğerinden farklı olacaktır.

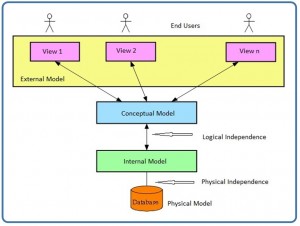
Dışsal model, tasarımcının gereksinim ve kısıtlama kümesini, dışsal modeller (örneğin insan kaynakları ile satışlar) çerçevesinde incelenebilecek işlevsel modüllere bölmesini gerektirir.

Bir veri tasarımcısı olarak, kuruluş çapında bir veritabanı oluşturabilmeniz için tüm verileri anlamanız gerekir. Çeşitli departmanların ihtiyaçlarına göre kavramsal model, oluşturulan ilk modeldir.

Bu aşamada, kavramsal model hem yazılımdan hem de donanımdan bağımsızdır. Modeli uygulamak için kullanılan DBMS yazılımına bağlı değildir. Modelin uygulanmasında kullanılan donanıma bağlı değildir. Donanım veya DBMS yazılımındaki değişikliklerin kavramsal düzeyde veritabanı tasarımı üzerinde hiçbir etkisi yoktur.

*Bir DBMS seçildikten sonra onu uygulayabilirsiniz. Bu dahili modeldir. Burada tüm tabloları, kısıtlamaları, anahtarları, kuralları vb. oluşturursunuz. Buna genellikle mantıksal tasarım* denir .

Fiziksel model, basitçe verilerin diskte depolanma şeklidir. Her veritabanı satıcısının verileri depolamanın kendine özgü bir yolu vardır.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Table-Description-300x146.jpg)Şekil 5.1. Veri soyutlama katmanları.

**Şemalar**

Bir *şema,* bir veritabanının genel bir tanımıdır ve genellikle  *varlık ilişkisi diyagramı (ERD)* ile temsil edilir . Harici modelleri temsil eden ve böylece verilerin harici görünümlerini görüntüleyen birçok alt şema vardır. Aşağıda, bir veritabanının tasarım süreci sırasında dikkate alınması gereken öğelerin bir listesi bulunmaktadır.

* Dış şemalar: birden fazla vardır
* Çoklu alt şemalar: bunlar verilerin çoklu dış görünümlerini görüntüler
* Kavramsal şema: yalnızca bir tane vardır. Bu şema, bir ERD'de temsil edilen veri öğelerini, ilişkileri ve kısıtlamaları içerir.
* Fiziksel şema: sadece bir tane var

**Mantıksal ve Fiziksel Veri Bağımsızlığı**

*Veri bağımsızlığı,* kullanıcı uygulamalarının veri tanımı ve organizasyonunda yapılan değişikliklere karşı bağışıklığı anlamına gelir. Veri soyutlamaları yalnızca kullanıcı için önemli veya alakalı olan öğeleri açığa çıkarır. Karmaşıklık veritabanı kullanıcısından gizlenir.

Veri bağımsızlığı ve işlem bağımsızlığı birlikte veri soyutlamasının özelliğini oluşturur. İki tür veri bağımsızlığı vardır: mantıksal ve fiziksel.

Mantıksal veri bağımsızlığı

Mantıksal *şema,* bir evin mimari çizimlerine çok benzeyen, kağıt veya beyaz tahta üzerinde yapılmış bir veritabanının kavramsal tasarımıdır. *Harici şemayı*  veya kullanıcı görünümünü  değiştirmeden mantıksal şemayı değiştirme becerisine *mantıksal veri bağımsızlığı*  denir . Örneğin, bu *kavramsal şemaya* yeni varlıklar, öznitelikler veya ilişkiler eklemek veya kaldırmak, mevcut harici şemaları değiştirmek veya mevcut uygulama programlarını yeniden yazmak zorunda kalmadan mümkün olmalıdır. 

Başka bir deyişle, mantıksal şemada yapılan değişiklikler (örneğin, veritabanının yapısında sütun veya diğer tabloların eklenmesi gibi değişiklikler) uygulamanın işlevini (dış görünümleri) etkilememelidir.

Fiziksel veri bağımsızlığı

*Fiziksel veri bağımsızlığı,* dahili modelin fiziksel modeldeki değişikliklere karşı bağışıklığını ifade eder. Dosya organizasyonunda veya depolama yapılarında, depolama aygıtlarında veya dizinleme stratejisinde değişiklikler yapılsa bile mantıksal şema değişmeden kalır.

Fiziksel veri bağımsızlığı, depolama yapısının ayrıntılarını kullanıcı uygulamalarından gizlemekle ilgilidir. Uygulamalar, verilere karşı gerçekleştirilen işlemde bir fark olmadığından bu sorunlarla ilgilenmemelidir.

Anahtar Terimler

**kavramsal model** :  tüm veritabanının mantıksal yapısı

**kavramsal şema** : mantıksal şema için başka bir terim

**veri bağımsızlığı** : kullanıcı uygulamalarının veri tanımı ve organizasyonunda yapılan değişikliklere karşı bağışıklığı

**veri modeli** : veriyi, veri ilişkilerini, veri semantiğini ve veri kısıtlamalarını tanımlamak için kullanılan kavram veya gösterimlerin bir koleksiyonu

**Veri modelleme** : Veritabanı tasarım sürecinin ilk adımı

**veritabanı mantıksal tasarımı** : belirli bir veritabanı yönetim sisteminin veri modelinde bir veritabanını tanımlar

**veritabanı fiziksel tasarımı *:***  dahili veritabanı depolama yapısını, dosya organizasyonunu veya dizinleme tekniklerini tanımlar

**varlık ilişki diyagramı (ERD)** : tabloları, öznitelikleri ve ilişkileri gösteren veritabanını tanımlayan bir veri modeli

**dış model** : kullanıcının veritabanına ilişkin görünümünü temsil eder

**harici şema** : kullanıcı görünümü

**dahili model:** DBMS tarafından görüldüğü gibi veritabanının bir temsili

**mantıksal veri bağımsızlığı** : harici şemayı değiştirmeden mantıksal şemayı değiştirme yeteneği

**mantıksal tasarım** : tüm tabloları, kısıtlamaları, anahtarları, kuralları vb. oluşturduğunuz yer

**mantıksal şema** : bir evin mimari çizimlerine çok benzeyen, kağıt veya beyaz tahta üzerinde yapılmış veritabanının kavramsal tasarımı

**işletim sistemi (OS)** : fiziksel modelin fiziksel düzeyini yönetir

**fiziksel veri bağımsızlığı** : dahili modelin fiziksel modeldeki değişikliklere karşı bağışıklığı

**fiziksel model** :  veritabanının fiziksel temsili

**şema** : bir veritabanının genel açıklaması

**İlişkisel Veri Modeli**

İlişkisel veri modeli 1970 yılında EF Codd tarafından ortaya atılmıştır. Günümüzde en yaygın kullanılan veri modelidir.

İlişkisel model aşağıdakilerin temelini oluşturmuştur:

* Veri/ilişki/kısıtlama teorisi üzerine araştırma
* Çok sayıda veritabanı tasarım metodolojisi
* *Yapılandırılmış sorgu dili (SQL)* olarak adlandırılan standart veritabanı erişim dili
* Hemen hemen tüm modern ticari veritabanı yönetim sistemleri

İlişkisel veri modeli dünyayı "birbiriyle ilişkili ilişkilerin (veya tabloların) bir koleksiyonu" olarak tanımlar.

**İlişkisel Veri Modelindeki Temel Kavramlar**

İlişki

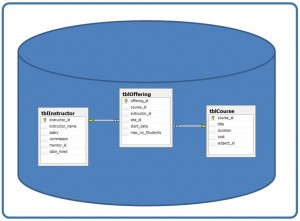
Bir  *ilişki,  tablo* veya *dosya* olarak da bilinir , bir adla karakterize edilen etki alanlarının listesinin Kartezyen ürününün bir alt kümesidir. Ve bir tablo içinde, her satır ilişkili veri değerlerinin bir grubunu temsil eder. Bir *satır veya kayıt, bir tuple* olarak da bilinir . Bir tablodaki sütunlar bir alandır ve bir öznitelik olarak da adlandırılır. Bunu şu şekilde de düşünebilirsiniz: bir öznitelik kaydı tanımlamak için kullanılır ve bir kayıt bir öznitelik kümesi içerir.

Aşağıdaki adımlar bir ilişki ile onun etki alanları arasındaki mantığı özetlemektedir.

1. Verilen *n*  etki alanı D1, D2, … Dn ile gösterilir
2. Ve *r,*  bu etki alanları üzerinde tanımlanmış bir ilişkidir
3. O zaman r ⊆ D1×D2×…×Dn

Masa

Bir veritabanı birden fazla tablodan oluşur ve her tablo verileri tutar. Şekil 7.1 üç tablo içeren bir veritabanını göstermektedir.

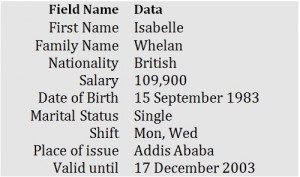
[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Banking-Systems-RDBMS-300x195.jpg)Şekil 7.1. Üç tablodan oluşan veritabanı.

Kolon

Bir veritabanı, bilgi veya olgu parçalarını düzenli bir şekilde depolar. Veritabanlarının nasıl kullanılacağını ve en iyi şekilde nasıl yararlanılacağını anlamak, bu organizasyon yöntemini anlamamızı gerektirir.

Başlıca depolama birimleri *sütunlar*  , *alanlar*  veya *öznitelikler* olarak adlandırılır . Bunlar, içeriğinizin parçalanabileceği temel veri bileşenlerini barındırır. Hangi alanları oluşturacağınıza karar verirken, bilgileriniz hakkında genel olarak düşünmeniz gerekir; örneğin, veritabanında depolayacağınız bilgilerin ortak bileşenlerini çizip bir öğeyi diğerinden ayıran ayrıntılardan kaçınmalısınız.

Alanlar ve bu alanlara ait veriler arasındaki ilişkiyi görmek için Şekil 7.2'deki kimlik kartı örneğine bakın.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Record-300x177.jpg)Şekil 7.2. A. Watt'ın kimlik kartı örneği.

İhtisas

Bir *alan,* verileri modellemek için kullanılan orijinal atomik değer kümeleridir. *Atomik değer* derken , alandaki her değerin ilişkisel model açısından bölünemez olduğunu kastediyoruz. Örneğin:

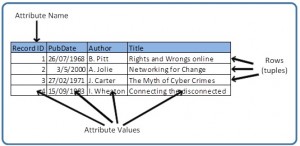
* Medeni Durum alanı bir dizi olasılık içerir: Evli, Bekar, Boşanmış.
* Shift'in etki alanı tüm olası günlerin kümesini içerir: {Pzt, Sal, Çar…}.
* Maaş alanı 0'dan büyük ve 200.000'den küçük tüm kayan nokta sayılarının kümesidir.
* Ad alanı, kişilerin adlarını temsil eden karakter dizilerinin kümesidir.

Özetle, bir etki alanı, bir sütunun içermesine izin verilen kabul edilebilir değerler kümesidir. Bu, çeşitli özelliklere ve sütun için veri türüne dayanır. Veri türlerini başka bir bölümde ele alacağız.

Kayıtlar

Herhangi bir belgenin veya öğenin içeriğinin, alanlarda depolanmak üzere bileşen veri parçalarına ayrılması gerektiği gibi, aralarındaki bağlantının da kullanılabilir olması gerekir, böylece bütün formlarına yeniden oluşturulabilirler. Kayıtlar bunu yapmamızı sağlar. *Kayıtlar,* müşteri veya çalışan gibi ilişkili alanlar içerir. Daha önce belirtildiği gibi, bir tuple, kayıt için kullanılan başka bir terimdir.

Kayıtlar ve alanlar tüm veritabanlarının temelini oluşturur. Basit bir tablo bize kayıtların ve alanların bir veritabanı depolama projesinde nasıl birlikte çalıştığına dair en net resmi verir.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/RDBMS-300x2091.jpg)Şekil 7.3. A. Watt'ın basit bir tablosunun örneği.

Şekil 7.3'teki basit tablo örneği bize alanların nasıl farklı türde veri aralığını tutabileceğini gösterir. Bu örnekte şunlar vardır:

* Kayıt Kimliği alanı: Bu bir sıra sayısıdır; veri türü tam sayıdır.
* PubDate alanı: Bu alan gün/ay/yıl olarak görüntülenir; veri türü tarihtir.
* Yazar alanı: Bu, Baş Harf. Soyadı olarak görüntülenir; veri türü metindir.
* Başlık alanı metni: Buraya serbest metin girilebilir.

Veritabanına verilerini elemesini ve belirli bir şekilde düzenlemesini emredebilirsiniz. Örneğin, kayıtların bir seçiminin tarihe göre sınırlandırılmasını isteyebilirsiniz: 1. belirli bir tarihten önceki tüm kayıtlar, 2. belirli bir tarihten sonraki tüm kayıtlar veya 3. belirli iki tarih arasındaki tüm kayıtlar. Benzer şekilde, kayıtların tarihe göre sıralanmasını seçebilirsiniz. Verileri içeren alan veya kayıt bir Tarih alanı olarak ayarlandığından, veritabanı Tarih alanındaki bilgileri yalnızca eğik çizgilerle ayrılmış sayılar olarak değil, takvim sistemine göre sıralanması gereken tarihler olarak okur.

Derece

Derece *,* bir tablodaki öznitelik sayısıdır. Şekil 7.3'teki örneğimizde derece 4'tür.

**Bir Tablonun Özellikleri**

* Bir tablonun veritabanındaki diğer tüm tablolardan farklı bir adı vardır.
* Tekrarlanan satırlar yoktur; her satır farklıdır.
* Sütunlardaki girdiler atomiktir. Tablo tekrarlayan gruplar veya çok değerli öznitelikler içermez  .
* Sütunlardaki girişler, veri türlerine göre aynı etki alanındandır:
  + sayı (sayısal, tamsayı, ondalık sayı, küçük tamsayı,…)
  + karakter (dize)
  + tarih
  + mantıksal (doğru veya yanlış)
* Farklı veri tiplerini birleştiren işlemlere izin verilmez.
* Her niteliğin ayrı bir ismi vardır.
* Sütunların sırası önemsizdir.
* Satırların sırası önemsizdir.

Anahtar Terimler

**atomik değer** : ilişkisel model açısından etki alanındaki her değer bölünemezdir. **nitelik** : bir veritabanındaki temel depolama birimi

**sütun:***özniteliğe* bakın

**derece** : bir tabloda bulunan niteliklerin sayısı

**etki alanı** : verileri modellemek için kullanılan orijinal atomik değer kümeleri; bir sütunun içermesine izin verilen kabul edilebilir değerler kümesi

**alan** :  *özniteliğe bakın*

**dosya** : *ilişkiye bakın*

**kayıt** : ilişkili alanları içerir; *tuple'a bakın*

**ilişki** : bir adla karakterize edilen alanların listesinin Kartezyen çarpımının bir alt kümesi; tablo veya dosya için kullanılan teknik terim

**satır** : *tuple'a bakın*

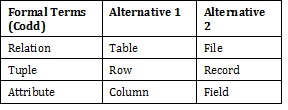
**Yapılandırılmış sorgu dili (SQL)** : Standart veritabanı erişim dili

**tablo:***ilişkiye* bakın

**tuple** : satır veya kayıt için kullanılan teknik bir terim

*Terminoloji Anahtarı*

Bu bölümde kullanılan terimlerin birçoğu eşanlamlıdır. Yukarıdaki Anahtar Terimlere ek olarak, lütfen aşağıdaki Tablo 7.1'e bakın. Alternatif 1 sütunundaki terimler en sık kullanılanlardır.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Terms.jpg)Tablo 7.1. A. Watt'ın terimleri ve eş anlamlıları.

**Varlık İlişkisi Veri Modeli**

Varlık *ilişkisi (ER) veri modeli* 35 yıldan uzun süredir mevcuttur. Veritabanlarıyla kullanım için veri modellemeye oldukça uygundur çünkü oldukça soyuttur ve tartışılması ve açıklanması kolaydır. ER modelleri ilişkilere kolayca çevrilebilir. ER modelleri, ER şeması olarak da adlandırılır ve ER diyagramlarıyla gösterilir.

ER modellemesi iki kavrama dayanmaktadır:

* *Belirli bilgileri (verileri )*  tutan tablolar olarak tanımlanan varlıklar
* *İlişkiler,*  varlıklar arasındaki ilişkiler veya etkileşimler olarak tanımlanır

Bu iki kavramın bir ER veri modelinde nasıl birleştirilebileceğine dair bir örnek şöyledir:  Prof. Ba (varlık) Veritabanı Sistemleri dersini (varlık ) vermektedir (ilişki ) .

Bu bölümün geri kalanında, ER modelinin kavramlarını göstermek için COMPANY veritabanı adı verilen bir örnek veritabanı kullanacağız. Bu veritabanı çalışanlar, departmanlar ve projeler hakkında bilgi içerir. Dikkat edilmesi gereken önemli noktalar şunlardır:

* Şirkette birkaç departman vardır. Her departmanın kendine özgü bir kimliği, bir adı, ofisin yeri ve departmanı yöneten belirli bir çalışanı vardır.
* Bir departman, her biri kendine özgü bir isme, numaraya ve bütçeye sahip çok sayıda projeyi kontrol eder.
* Her çalışanın bir adı, kimlik numarası, adresi, maaşı ve doğum tarihi vardır. Bir çalışan bir departmana atanır ancak birden fazla projeye katılabilir. Her projede çalışanın başlangıç ​​tarihini kaydetmemiz gerekir. Ayrıca her çalışanın doğrudan amirini de bilmemiz gerekir.
* Her çalışanın bakmakla yükümlü olduğu kişileri takip etmek istiyoruz. Her bakmakla yükümlü olunan kişinin bir adı, doğum tarihi ve çalışanla ilişkisi vardır.

**Varlık, Varlık Kümesi ve Varlık Türü**

Bir *varlık* , gerçek dünyada bağımsız bir varlığa sahip ve diğer nesnelerden ayırt edilebilen bir nesnedir. Bir varlık şu şekilde olabilir:

* Fiziksel varlığı olan bir nesne (örneğin, bir öğretim görevlisi, bir öğrenci, bir araba)
* Kavramsal varlığı olan bir nesne (örneğin, bir ders, bir iş, bir pozisyon)

Varlıklar güçlerine göre sınıflandırılabilir. Bir varlık, tabloları varoluşa bağlıysa zayıf olarak kabul edilir.

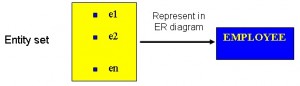
* Yani başka bir varlıkla ilişkisi olmadan var olamaz
* Birincil anahtarı, ana varlığın birincil anahtarından türetilir
  + COMPANY veritabanındaki Eş tablosu, birincil anahtarı Çalışan tablosuna bağlı olduğundan zayıf bir varlıktır. Karşılık gelen bir çalışan kaydı olmadan, eş kaydı var olmazdı.

Bir varlık, kendisine bağlı tüm varlıklardan ayrı olarak var olabiliyorsa güçlü kabul edilir.

* Çekirdekler güçlü varlıklardır.
* Yabancı anahtarı olmayan veya boş değerler içerebilen yabancı anahtar içeren bir tablo güçlü bir varlıktır

Bilmeniz gereken bir diğer terim ise benzer varlıkların bir koleksiyonunu tanımlayan *varlık türüdür .*

Bir *varlık kümesi,* belirli bir zaman noktasında bir varlık türünün varlıklarının bir koleksiyonudur. Bir varlık ilişkisi diyagramında (ERD), bir varlık türü bir kutudaki bir adla temsil edilir. Örneğin, Şekil 8.1'de varlık türü ÇALIŞAN'dır.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Attributes-300x106.jpg)Şekil 8.1. Çalışan varlık türüne sahip ERD.

Varoluş bağımlılığı

Bir varlığın varlığı, ilgili varlığın varlığına bağlıdır. Zorunlu bir yabancı anahtara (yani, boş olamayacak bir yabancı anahtar niteliğine) sahipse, varlığa bağlıdır. Örneğin, COMPANY veritabanında, bir Eş varlığı, Çalışan varlığına bağlıdır.

**Varlık Çeşitleri**

Bağımsız varlıklar, bağımlı varlıklar ve karakteristik varlıklar dahil olmak üzere farklı varlık türlerine de aşina olmalısınız. Bunlar aşağıda açıklanmıştır.

Bağımsız varlıklar

*Bağımsız varlıklar* , çekirdekler olarak da adlandırılır, veritabanının omurgasıdır. Diğer tabloların dayandığı şeydir. *Çekirdekler* aşağıdaki özelliklere sahiptir:

* Bunlar bir veritabanının yapı taşlarıdır.
* Birincil anahtar basit veya bileşik olabilir.
* Birincil anahtar yabancı anahtar değildir.
* Varlıkları için başka bir varlığa bağımlı değillerdir.

ŞİRKET veritabanımıza geri dönersek, bağımsız bir varlığın örnekleri arasında Müşteri tablosu, Çalışan tablosu veya Ürün tablosu bulunur.

Bağımlı varlıklar

*Bağımlı varlıklar , türetilmiş varlıklar* olarak da adlandırılır , anlamları için diğer tablolara bağlıdır. Bu varlıklar aşağıdaki özelliklere sahiptir:

* Bağımlı varlıklar iki çekirdeği birbirine bağlamak için kullanılır.
* İki veya daha fazla tabloya bağlı varoluşa sahip oldukları söylenir.
* Çoktan çoğa ilişkiler en az iki yabancı anahtarla ilişkisel tablolara dönüşür.
* Başka nitelikler de içerebilirler.
* Yabancı anahtar her ilişkili tabloyu tanımlar.
* Birincil anahtar için üç seçenek vardır:
  1. Benzersizse ilişkili tabloların yabancı anahtarlarının bir bileşimini kullanın
  2. Yabancı anahtarların ve niteleyici bir sütunun bileşimini kullanın
  3. Yeni basit bir birincil anahtar oluşturun

Karakteristik varlıklar

*Karakteristik varlıklar* başka bir tablo hakkında daha fazla bilgi sağlar. Bu varlıklar aşağıdaki özelliklere sahiptir:

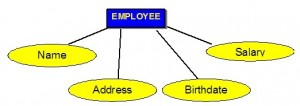
* Çok değerli nitelikleri temsil ederler.
* Başka varlıkları tarif ederler.
* Genellikle bire-çok ilişkisi vardır.
* Yabancı anahtar, karakterize edilen tabloyu daha ayrıntılı tanımlamak için kullanılır.
* Birincil anahtar için seçenekler aşağıdaki gibidir:
  1. Yabancı anahtar ve niteleyici sütunun bir bileşimini kullanın
  2. Yeni bir basit birincil anahtar oluşturun. COMPANY veritabanında bunlar şunları içerebilir:
     + Çalışan ( EID, İsim, Adres, Yaş, Maaş) – EID basit birincil anahtardır.
     + EmployeePhone ( EID, Phone ) – EID, bileşik birincil anahtarın bir parçasıdır. Burada, EID aynı zamanda bir yabancı anahtardır.

**Nitelikler**

Her varlık bir dizi öznitelikle tanımlanır (örneğin, Çalışan = (Ad, Adres, Doğum Tarihi (Yaş) , Maaş).

Her özniteliğin bir adı vardır ve bir varlık ve yasal değerlerin bir alanıyla ilişkilendirilir. Ancak, öznitelik alanıyla ilgili bilgiler ERD'de sunulmaz.

Şekil 8.2'de gösterilen varlık ilişkisi diyagramında, her nitelik, içinde bir ad bulunan bir oval ile temsil edilir.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Distributed-Systems-300x217.jpg)Şekil 8.2. Niteliklerin bir ERD'de nasıl temsil edildiği.

**Nitelik Türleri**

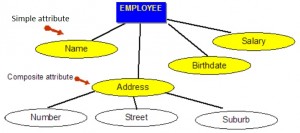
Bilmeniz gereken birkaç tür öznitelik vardır. Bunlardan bazıları olduğu gibi bırakılmalı, ancak bazılarının ilişkisel modelde temsili kolaylaştırmak için ayarlanması gerekir. Bu ilk bölümde öznitelik türleri ele alınacaktır. Daha sonra özniteliklerin ilişkisel modele doğru şekilde uyması için nasıl düzeltileceğini ele alacağız.

Basit nitelikler

*Basit nitelikler* atomik değer alanlarından çekilenlerdir; bunlara *tek değerli nitelikler* de denir . COMPANY veritabanında buna bir örnek şöyle olur: Name = {John} ; Age = {23}

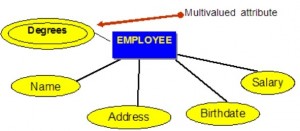
Bileşik nitelikler

*Bileşik nitelikler,* bir nitelikler hiyerarşisinden oluşan niteliklerdir. Veritabanı örneğimizi kullanarak ve Şekil 8.3'te gösterildiği gibi, Adres; Sayı, Sokak ve Banliyö'den oluşabilir. Bu nedenle bu şu şekilde yazılır: → Adres = {59 + 'Meek Street' + 'Kingsford'}

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Network-data-model-300x244.jpg)Şekil 8.3. Bileşik niteliklere bir örnek.

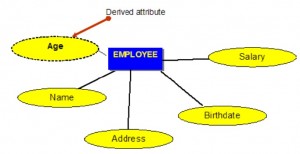
Çok değerli nitelikler

*Çok değerli nitelikler,* her varlık için bir değer kümesine sahip niteliklerdir. Şekil 8.4'te görüldüğü gibi, ŞİRKET veritabanından çok değerli bir niteliğe örnek olarak bir çalışanın dereceleri verilebilir: BSc, MIT, PhD.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Many-to-Many-Relationships-300x167.jpg)Şekil 8.4. Çok değerli bir özniteliğin örneği.

Türetilmiş nitelikler

*Türetilmiş nitelikler,*  diğer niteliklerden hesaplanan değerleri içeren niteliklerdir. Bunun bir örneği Şekil 8.5'te görülebilir. Yaş, Birthdate niteliğinden türetilebilir. Bu durumda,  Birthdate , veritabanına fiziksel olarak kaydedilen *depolanmış* bir nitelik olarak adlandırılır .

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Derived-Attribute-300x154.jpg)Şekil 8.5. Türetilmiş bir niteliğin örneği.

**Anahtarlar**

Bir varlık üzerindeki önemli bir kısıtlama anahtardır. *Anahtar,* değerleri bir varlık kümesindeki bireysel bir varlığı benzersiz şekilde tanımlamak için kullanılabilen bir öznitelik veya bir öznitelik grubudur.

**Anahtar Türleri**

Birkaç anahtar türü vardır. Bunlar aşağıda açıklanmıştır.

Aday anahtarı

Aday *anahtar,* benzersiz ve minimal olan basit veya bileşik bir anahtardır. Benzersizdir çünkü bir tablodaki iki satırın aynı anda aynı değere sahip olması mümkün değildir. Benzersizdir çünkü benzersizliği elde etmek için her sütun gereklidir.

ŞİRKET veritabanı örneğimizden, eğer varlık **Çalışan** (EID, Adı, Soyadı, *SIN* , Adres, Telefon, Doğum Tarihi, Maaş, Departman Kimliği) ise, olası aday anahtarlar şunlardır:

* Bayram, Günah
* Ad ve Soyad – Şirkette aynı adı taşıyan başka kimse olmadığını varsayarak
* Soyadı ve Departman Kimliği – aynı soyadına sahip iki kişinin aynı departmanda çalışmadığını varsayarak

Bileşik anahtar

Bileşik *anahtar* iki veya daha fazla öznitelikten oluşur, ancak minimum düzeyde olmalıdır.

Aday anahtar bölümündeki örneği kullanarak olası bileşik anahtarlar şunlardır:

* Ad ve Soyad – Şirkette aynı adı taşıyan başka kimse olmadığını varsayarak
* Soyadı ve Departman Kimliği – aynı soyadına sahip iki kişinin aynı departmanda çalışmadığını varsayarsak

Birincil anahtar

Birincil anahtar, veritabanı tasarımcısı tarafından tüm varlık kümesi için bir tanımlama mekanizması olarak kullanılmak üzere seçilen aday bir anahtardır. Bir tabloda bulunan tuple'ları benzersiz bir şekilde tanımlamalı ve boş olmamalıdır. Birincil anahtar, ER modelinde özniteliğin altını çizerek belirtilir.

* Bir aday anahtar, bir tabloda bulunan tuple'ları benzersiz bir şekilde tanımlamak için tasarımcı tarafından seçilir . Boş olmamalıdır.
* Veritabanı tasarımcısı tarafından tüm varlık kümesi için bir tanımlama mekanizması olarak kullanılacak bir anahtar seçilir. Buna birincil anahtar denir. Bu anahtar, ER modelinde özniteliğin altını çizerek belirtilir.

Aşağıdaki örnekte EID birincil anahtardır:

**Çalışan** ( EID , Adı, Soyadı, SIN, Adres, Telefon, Doğum Tarihi, Maaş, Departman Kimliği)

İkincil anahtar

İkincil *anahtar* , yalnızca bilgi alma amacıyla kullanılan bir niteliktir (bileşik olabilir), örneğin: Telefon ve Soyadı.

Alternatif anahtar

*Alternatif anahtarlar,*  birincil anahtar olarak seçilmeyen aday anahtarlardır.

Yabancı anahtar

Yabancı anahtar *(FK),* başka bir tablodaki birincil anahtara başvuran bir tablodaki bir özniteliktir VEYA boş olabilir. Hem yabancı hem de birincil anahtarlar aynı veri türünde olmalıdır.

Aşağıdaki ŞİRKET veritabanı örneğinde, DepartmentID yabancı anahtardır:

**Çalışan** (EID, Adı, Soyadı, SIN, Adres, Telefon, Doğum Tarihi, Maaş, Departman Kimliği)

**Boşlar**

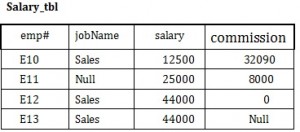
Null *,* veri türünden bağımsız, bilinmeyen veya uygulanamaz anlamına gelen özel bir semboldür. Sıfır veya boş anlamına gelmez. Null'un özellikleri şunlardır:

* Veri girişi yok
* Birincil anahtarda izin verilmez
* Diğer niteliklerde kaçınılmalıdır
* Temsil edebilir
  + Bilinmeyen bir öznitelik değeri
  + Bilinen ancak eksik bir öznitelik değeri
  + "Uygulanamaz" koşulu
* COUNT, AVERAGE ve SUM gibi fonksiyonlar kullanıldığında sorunlara yol açabilir
* İlişkisel tablolar birbirine bağlandığında mantıksal sorunlara yol açabilir

NOT: Bir karşılaştırma işleminin sonucu, her iki argüman da null olduğunda null olur. Bir aritmetik işleminin sonucu, her iki argüman da null olduğunda null olur (null'ları yoksayan işlevler hariç).

null'ın nasıl kullanılabileceğine dair bir örnek

Şekil 8.6'daki Maaş tablosunu (Salary\_tbl) kullanarak null değerinin nasıl kullanılabileceğine dair bir örnek izleyin.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/SalaryTable-300x132.jpg)Şekil 8.6. A. Watt tarafından hazırlanan boş örnek için maaş tablosu.

Başlamak için Satış bölümünde (jobName sütununun altında) maaşı artı komisyonu 30.000'den fazla olan tüm çalışanları (emp#) bulun.

* Salary\_tbl'den emp#'yi SEÇ
* WHERE jobName = Satışlar VE
* (komisyon + maaş) > 30.000 –> E10 ve E12

Bu sonuç, komisyon sütunundaki boş değer nedeniyle E13'ü içermez. Boş değere sahip satırın dahil edildiğinden emin olmak için, bireysel alanlara bakmamız gerekir. Çalışan E13 için komisyon ve maaş eklendiğinde, sonuç boş bir değer olacaktır. Çözüm aşağıda gösterilmiştir.

* Salary\_tbl'den emp#'yi SEÇ
* WHERE jobName = Satışlar VE
* (komisyon > 30000 VEYA
* maaş > 30000 VEYA
* (komisyon + maaş) > 30.000 –>E10 ve E12 ve E13

**İlişkiler**

*İlişkiler* , tabloları bir arada tutan yapıştırıcıdır. Tablolar arasındaki ilgili bilgileri birbirine bağlamak için kullanılırlar .

*İlişki gücü,* ilgili bir varlığın birincil anahtarının nasıl tanımlandığına dayanır. İlgili varlığın birincil anahtarı, ana varlığın birincil anahtar bileşenini içermiyorsa zayıf veya tanımlayıcı olmayan bir ilişki vardır. Şirket veritabanı örnekleri şunları içerir:

* Müşteri( **MüşteriKimliği** , MüşteriAdı)
* Sipariş( **SiparişKimliği** , MüşteriKimliği, Tarih)

İlgili varlığın birincil anahtarı, ana varlığın birincil anahtar bileşenini içerdiğinde güçlü veya tanımlayıcı bir ilişki vardır. Örnekler şunları içerir:

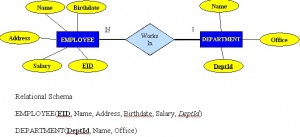
* Ders( **CrsCode** , DeptCode, Açıklama)
* Sınıf( **CrsCode, Section** , ClassTime…)

**İlişki Türleri**

Aşağıda çeşitli ilişki türlerinin açıklamaları yer almaktadır.

Birden çoğa (1:M) ilişki

Birden çoğa (1:M) ilişki, herhangi bir ilişkisel veritabanı tasarımında norm olmalıdır ve tüm ilişkisel veritabanı ortamlarında bulunur. Örneğin, bir departmanın birçok çalışanı vardır. Şekil 8.7, bu çalışanlardan birinin departmanla ilişkisini gösterir.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/FileBased-300x170.jpg)Şekil 8.7. Bire çok ilişkisine örnek.

Birebir (1:1) ilişki

Bire bir (1:1) ilişki, bir varlığın yalnızca bir diğer varlığa ve tam tersine olan ilişkisidir. Herhangi bir ilişkisel veritabanı tasarımında nadir olmalıdır. Aslında, iki varlığın aslında aynı tabloya ait olduğunu gösterebilir.

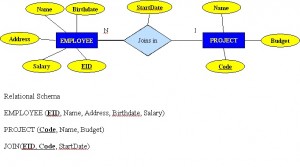
ŞİRKET veritabanından bir örnek; bir çalışanın bir eşle, bir eşin de bir çalışanla ilişkili olmasıdır.

Çoktan çoğa  (M:N) ilişkiler

Çoktan çoğa ilişki için aşağıdaki noktaları göz önünde bulundurun:

* İlişkisel modelde bu şekilde uygulanması mümkün değildir.
* İki adet 1:M ilişkisine dönüştürülebilir.
* 1:M ilişkilerinden oluşan bir küme üretmek için parçalanarak uygulanabilir.
* Bileşik bir varlığın uygulanmasını içerir .
* İki veya daha fazla 1:M ilişkisi oluşturur.
* Bileşik varlık tablosu en azından orijinal tabloların birincil anahtarlarını içermelidir.
* Bağlantı tablosu yabancı anahtar değerlerinin birden fazla örneğini içerir.
* İhtiyaç halinde ek nitelikler atanabilir.
* Bir bileşik varlık veya köprü varlığı oluşturarak M:N ilişkisinde bulunan sorunları önleyebilir. Örneğin, bir çalışan birçok proje üzerinde çalışabilir VEYA bir projede, iş kurallarına bağlı olarak birçok çalışan çalışabilir. Veya bir öğrencinin birçok sınıfı ve bir sınıfın birçok öğrencisi olabilir.

Şekil 8.8, bir çalışanın farklı projeler için farklı başlangıç ​​tarihlerine sahip olduğu M:N ilişkisinin başka bir yönünü gösterir . Bu nedenle, EID, Kod ve Başlangıç ​​Tarihini içeren bir JOIN tablosuna ihtiyacımız var.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/One-to-Many-Relationships-300x137.jpg)Şekil 8.8. Çalışanın farklı projeler için farklı başlangıç ​​tarihlerine sahip olduğu örnek.

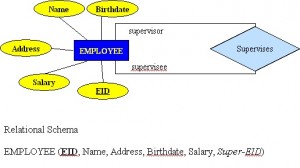
**M:N ikili ilişki türünün eşlenmesine örnek**

* Her M:N ikili ilişkisi için iki ilişki tanımlayın.
* A ve B, R'ye katılan iki varlık türünü temsil eder.
* R'yi temsil edecek yeni bir S ilişkisi yaratın.
* S, A ve B'nin PK'larını içermelidir. Bunlar birlikte S tablosundaki PK olabilir VEYA bunlar yeni tablo R'deki başka bir basit nitelikle birlikte PK olabilir.
* Birincil tuşların (A ve B) birleşimi S'nin birincil tuşunu oluşturacaktır.

Tekli ilişki (tekrarlı)

Tekli *ilişki, aynı* varlık kümesinin oluşumları arasında bir ilişkinin var olduğu ilişkidir . Bu ilişkide, birincil ve yabancı anahtarlar aynıdır, ancak farklı rollere sahip iki varlığı temsil ederler. Bir örnek için Şekil 8.9'a bakın.

Tekli ilişkideki bazı varlıklar için , aynı varlık kümesinin birincil anahtarına başvuran ayrı bir sütun oluşturulabilir.

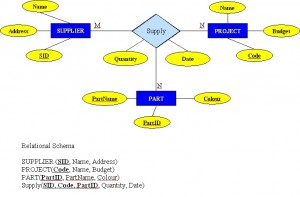
[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Data-Abstraction-300x226.jpg)Şekil 8.9. Tekli ilişkiye örnek.

Üçlü İlişkiler

Üçlü *ilişki,* üç tablo arasında çoktan çoğa ilişkileri içeren bir ilişki türüdür.

Üçlü ilişki tipinin eşlenmesine dair bir örnek için Şekil 8.10'a bakın. Not:  *n-ary,* bir ilişkideki birden fazla tablo anlamına gelir. (Unutmayın, N = çok.)

* Her n-ary (> 2) ilişki için, ilişkiyi temsil edecek yeni bir ilişki oluşturun.
* Yeni ilişkinin birincil anahtarı, N (çok) tarafını tutan katılımcı varlıkların birincil anahtarlarının birleşimidir.
* n-li ilişkilerin çoğu durumunda, katılan tüm varlıklar **çok** tarafına sahiptir.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ternary-Mapping-Relationships-300x197.jpg)Şekil 8.10. Üçlü ilişkiye örnek.

Anahtar Terimler

**alternatif anahtar** : birincil anahtar olarak seçilmeyen tüm aday anahtarlar **aday anahtar** : benzersiz (bir tablodaki iki satır aynı değere sahip olamaz) ve minimum (her sütun gereklidir) olan basit veya bileşik bir anahtar

**karakteristik varlıklar** : başka bir tablo hakkında daha fazla bilgi sağlayan varlıklar

**bileşik nitelikler** : niteliklerin hiyerarşisinden oluşan nitelikler

**bileşik anahtar** : iki veya daha fazla öznitelikten oluşur, ancak en az düzeyde olmalıdır

**bağımlı varlıklar** : bu varlıklar anlamları için diğer tablolara bağlıdır

**türetilmiş nitelikler** : diğer niteliklerden hesaplanan değerleri içeren nitelikler

**türetilmiş varlıklar** : *bağımlı varlıklara bakın*

**EID** : çalışan kimliği (ID)

**varlık** : gerçek dünyada bağımsız bir varoluşa sahip olan ve diğer nesnelerden ayırt edilebilen bir şey veya nesne

**varlık ilişkisi (ER) veri modeli** :  ER şeması olarak da adlandırılır , ER diyagramlarıyla gösterilir. Bunlar veritabanlarıyla kullanım için veri modellemeye çok uygundur.

**varlık ilişkisi şeması**: *varlık ilişkisi veri modeline bakın*

**varlık kümesi** : belirli bir zamanda bir varlık türünün varlıklarının koleksiyonu

**varlık türü** : benzer varlıkların bir koleksiyonu

**yabancı anahtar (FK)** : başka bir tablodaki birincil anahtara başvuran bir tablodaki bir öznitelik VEYA boş olabilir

**bağımsız varlık** : bir veritabanının yapı taşları olarak, bu varlıklar diğer tabloların dayandığı varlıklardır

**çekirdek :***bağımsız varlığa* bakın

**anahtar** : değerleri bir varlık kümesindeki belirli bir varlığı benzersiz şekilde tanımlamak için kullanılabilen bir nitelik veya nitelik grubu

**çok değerli nitelikler** : her varlık için bir değer kümesine sahip nitelikler

**n-ary** : bir ilişkideki birden fazla tablo

**null** : veri türünden bağımsız, bilinmeyen veya uygulanamaz anlamına gelen özel bir sembol; sıfır veya boş anlamına gelmez

**yinelemeli ilişki** : *tekli ilişkiye bakınız*

**ilişkiler** : varlıklar arasındaki ilişkiler veya etkileşimler; tablolar arasında ilgili bilgileri birbirine bağlamak için kullanılır

**ilişki gücü** : ilgili bir varlığın birincil anahtarının nasıl tanımlandığına bağlı

**ikincil anahtar,** yalnızca bilgi alma amaçları için kullanılan bir nitelik

**basit nitelikler** : atomik değer alanlarından elde edilir

**SIN** : sosyal sigorta numarası

**tek değerli nitelikler** : *basit niteliklere bakın*

**saklanan öznitelik** : fiziksel olarak veritabanına kaydedildi

**üçlü ilişki** : üç tablo arasında çoktan çoğa ilişki içeren bir ilişki türü.

**tekli ilişki** : aynı varlık kümesinin oluşumları arasında bir ilişkinin var olduğu ilişki.

Dürüstlük Kuralları ve Kısıtlamaları

Kısıtlamalar ilişkisel bir modelde çok önemli bir özelliktir. Aslında, ilişkisel model nitelikler veya tablolar üzerindeki iyi tanımlanmış kısıtlamalar teorisini destekler. Kısıtlamalar yararlıdır çünkü bir tasarımcının veritabanındaki verilerin semantiğini belirlemesine olanak tanır. Kısıtlamalar, *DBMS'leri* verilerin semantiği karşıladığını kontrol etmeye zorlayan kurallardır.

**Alan Bütünlüğü**

Alan, ilişkideki öznitelik değerlerini kısıtlar ve ilişkisel modelin bir kısıtlamasıdır. Ancak, yalnızca alan kısıtlamalarıyla kullanıldığında belirtilemeyen veriler için gerçek dünya semantiği vardır. Hangi veri değerlerinin izin verilip verilmediğini ve bir öznitelik için hangi biçimin uygun olduğunu belirtmek için daha belirli yollara ihtiyacımız var. Örneğin, Çalışan Kimliği (EID) benzersiz olmalı veya çalışan Doğum Tarihi [1 Ocak 1950, 1 Ocak 2000] aralığında olmalıdır. Bu tür bilgiler *bütünlük kısıtlamaları* adı verilen mantıksal ifadelerde sağlanır .

Aşağıda açıklanan çeşitli bütünlük kısıtlamaları vardır.

Kuruluş bütünlüğü

*Varlık bütünlüğünü* sağlamak için her tablonun birincil bir anahtara sahip olması gerekir. Ne PK ne de herhangi bir parçası boş değerler içeremez. Bunun nedeni, birincil anahtar için boş değerlerin bazı satırları tanımlayamayacağımız anlamına gelmesidir. Örneğin, EMPLOYEE tablosunda, bazı kişilerin telefonu olmayabileceği için Telefon birincil anahtar olamaz.

Referans bütünlüğü

*Referans bütünlüğü,* yabancı anahtarın eşleşen bir birincil anahtara sahip olması veya boş olması gerektiğini gerektirir. Bu kısıtlama iki tablo (üst ve alt) arasında belirtilir; bu tablolardaki satırlar arasındaki yazışmayı korur. Bu, bir tablodaki bir satırdan başka bir tabloya yapılan referansın geçerli olması gerektiği anlamına gelir.

Şirketin Müşteri/Sipariş veritabanında referans bütünlüğü kısıtlamasına ilişkin örnekler:

* Müşteri( **MüşteriKimliği** , MüşteriAdı)
* Sipariş( **SiparişKimliği** , MüşteriKimliği, SiparişTarihi)

Yetim kayıt olmadığından emin olmak için referans bütünlüğünü zorunlu kılmamız gerekir. *Yetim kayıt,* yabancı anahtar FK değeri karşılık gelen varlıkta (PK'nin bulunduğu varlık) bulunmayan kayıttır. Tipik bir birleşimin bir PK ile FK arasında olduğunu hatırlayın.

Referans bütünlüğü kısıtlaması, Order tablosundaki müşteri kimliğinin (CustID) Customer tablosundaki geçerli bir CustID ile eşleşmesi gerektiğini belirtir. Çoğu ilişkisel veritabanının bildirimsel referans bütünlüğü vardır. Başka bir deyişle, tablolar oluşturulduğunda referans bütünlüğü kısıtlamaları ayarlanır.

İşte Ders/Sınıf veritabanından bir örnek daha:

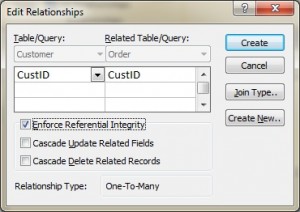
* Ders( **CrsCode** , DeptCode, Açıklama)
* Sınıf( **CrsCode, Bölüm** , SınıfZamanı)

Referans bütünlüğü kısıtlaması, Class tablosundaki CrsCode'un Course tablosundaki geçerli bir CrsCode ile eşleşmesi gerektiğini belirtir. Bu durumda, Class tablosundaki CrsCode ve Section'ın PK'yi oluşturması yeterli değildir , ayrıca referans bütünlüğünü de zorunlu kılmalıyız.

Referans bütünlüğünü ayarlarken PK ve FK'nin aynı veri türlerine sahip olması ve aynı etki alanından gelmesi önemlidir , aksi takdirde ilişkisel veritabanı yönetim sistemi (RDBMS) birleştirmeye izin vermez. RDBMS, IBM'in San Jose Araştırma Laboratuvarı'ndan EF Codd tarafından tanıtılan ilişkisel modele dayanan popüler bir veritabanı sistemidir. İlişkisel veritabanı sistemleri diğer veritabanı sistemlerine göre kullanımı ve anlaşılması daha kolaydır.

Microsoft Access'te referans bütünlüğü

Microsoft (MS) Access'te, referans bütünlüğü, Müşteri tablosundaki PK'yi Sipariş tablosundaki Müşteri Kimliği'ne ekleyerek kurulur . Bunun MS Access'teki İlişkileri Düzenle ekranında nasıl yapıldığına dair bir görünüm için Şekil 9.1'e bakın.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Access-Referential-Integrity-Window-300x212.jpg)Şekil 9.1. A. Watt tarafından MS Access'te referans erişimi.

Transact-SQL (MS SQL Server) kullanarak referans bütünlüğü

Transact-SQL kullanıldığında, referans bütünlüğü FK ile Order tablosunu oluştururken ayarlanır . Aşağıda, Order tablosundaki FK'nin Customer tablosundaki PK'ye referans verdiğini gösteren ifadeler listelenmiştir.

Müşteri Tablosu Oluştur  
( Müşteri Kimliği Tam Sayı Birincil Anahtar,  
Müşteri Adı CHAR(35) )

Siparişler tablosunu oluştur  
(SiparişKimliği TAM SAYI BİRİNCİL ANAHTAR,  
MüşteriKimliği TAM SAYI BAŞVURULARI Müşteri(MüşteriKimliği),  
SiparişTarihi TARİHSAAT)

Yabancı anahtar kuralları

Referans bütünlüğünü ayarlarken ek yabancı anahtar kuralları eklenebilir, örneğin PK'li kayıt, ana (Müşteri) parçası olduğunda, alt satırlarla (Siparişler tablosunda) ne yapılacağı gibi. Örneğin, MS Access'teki İlişkileri Düzenle penceresi (bkz. Şekil 9.1) FK kuralları için iki ek seçenek gösterir: Basamaklı Güncelleme ve Basamaklı Silme. Bunlar seçilmezse, bir alt kayıt varsa sistem ana tablodaki (Müşteri tablosu) PK değerlerinin silinmesini veya güncellenmesini engeller. Alt kayıt, eşleşen bir PK'ye sahip herhangi bir kayıttır.

Bazı veritabanlarında, Sil seçeneği seçildiğinde Set to Null adlı ek bir seçenek bulunur. Bu seçildiğinde, PK satırı silinir, ancak alt tabloda FK NULL olarak ayarlanır. Bu bir yetim satır oluştursa da kabul edilebilir.

**Kurumsal Kısıtlamalar**

İşletme kısıtlamaları (bazen anlamsal kısıtlamalar olarak da adlandırılır) kullanıcılar veya veritabanı yöneticileri tarafından belirtilen ek kurallardır ve birden fazla tabloya dayalı olabilir.

İşte birkaç örnek.

* Bir sınıfta en fazla 30 öğrenci bulunabilir.
* Bir öğretmen bir yarıyılda en fazla dört derse ders verebilir.
* Bir çalışan beşten fazla projede yer alamaz.
* Bir çalışanın maaşı, çalışanın yöneticisinin maaşından fazla olamaz.

**İş Kuralları**

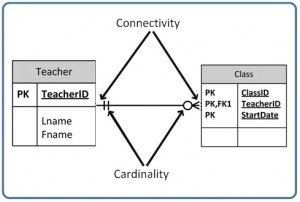
*İş kuralları,* gereksinimler toplanırken kullanıcılardan elde edilir. Gereksinim toplama süreci çok önemlidir ve sonuçları veritabanı tasarımı oluşturulmadan önce kullanıcı tarafından doğrulanmalıdır. İş kuralları yanlışsa, tasarım da yanlış olur ve sonuçta oluşturulan uygulama kullanıcıların beklediği gibi çalışmaz.

İş kurallarına bazı örnekler şunlardır:

* Bir öğretmen birçok öğrenciye ders verebilir.
* Bir sınıfta en fazla 35 öğrenci bulunabilir.
* Bir ders birçok kez verilebilir, ancak tek bir eğitmen tarafından.
* Bütün öğretmenler ders vermez.

Kardinalite ve bağlantı

Kardinalite ve bağlantıyı belirlemek için iş kuralları kullanılır . *Kardinalite,* iki veri tablosu arasındaki ilişkiyi, ilgili bir varlığın bir oluşumuyla ilişkili minimum ve maksimum varlık oluşum sayısını ifade ederek açıklar. Şekil 9.2'de, kardinalitenin ilişki simgesindeki en içteki işaretlerle temsil edildiğini görebilirsiniz. Bu şekilde, kardinalite sağda 0 (sıfır) ve solda 1'dir (bir).

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Connectivity-and-Cardinality-300x202.jpg)Şekil 9.2. A. Watt'ın bir ilişki sembolünde bağlantı ve kardinalite konumu.

İlişki sembolünün en dıştaki sembolü ise iki tablo arasındaki bağlantıyı temsil eder. *Bağlantı* , iki tablo arasındaki ilişkidir, örneğin, bire bir veya bire çok . Sıfır olduğu tek zaman, FK'nin boş olabileceği zamandır. Katılım söz konusu olduğunda, bu varlıklar arasındaki  ilişkiye ilişkin  üç seçenek vardır : 0 (sıfır), 1 (bir) veya çok.  Örneğin, Şekil 9.2'de, bağlantı bu çizginin dış, sol tarafında 1 (bir) ve dış, sağ tarafında çoktur.

Şekil 9.3, bire çok ilişkiyi temsil eden sembolü göstermektedir.

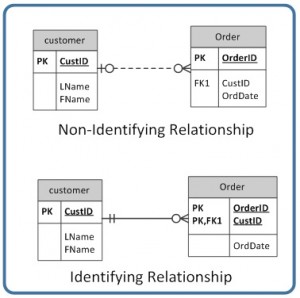
[Ch-9-birdençoğa-1](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-oneToMany-1.jpg)Şekil 9.3.

Şekil 9.4'te hem iç (kardinaliteyi temsil eden) hem de dış (bağlantıyı temsil eden) işaretleyiciler gösterilmiştir. Bu sembolün sol tarafı minimum 1 ve maksimum 1 olarak okunur. Sağ tarafta ise minimum 1 ve maksimum çok olarak okunur.

[Bölüm-9-birdençoğa-2](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-oneToMany-2.jpg)Şekil 9.4.

**İlişki Türleri**

Bir ERD'de iki tabloyu birbirine bağlayan çizgi, tablolar arasındaki *ilişki türünü belirtir :* tanımlayıcı veya tanımlayıcı olmayan. *Tanımlayıcı bir ilişki,* düz bir çizgiye sahip olacaktır (PK'nin FK'yi içerdiği yer). *Tanımlayıcı olmayan bir ilişki, kesik* bir çizgiyle gösterilir ve PK'de FK'yi içermez. Daha fazla açıklama için Bölüm 8'deki zayıf ve güçlü ilişkileri ele alan bölüme bakın.

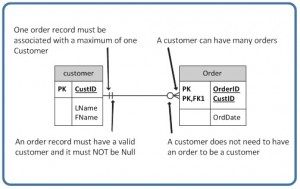
[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-Identifying-and-Non-Identifying-relationship-300x298.jpg)Şekil 9.5. Tanımlayıcı ve tanımlayıcı olmayan ilişki, A. Watt.

İsteğe bağlı ilişkiler

*İsteğe bağlı* bir ilişkide , FK boş olabilir veya ana tablonun karşılık gelen bir alt tablo oluşumuna sahip olması gerekmez  . Şekil 9.6'da gösterilen sembol, sıfır ve üç çatallı (çok olduğunu gösterir) bir türü gösterir ve bu sıfır VEYA çok olarak yorumlanır.

[Ch-9-Sıfır-veya-Çok-1](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-Zero-or-Many-1.jpg)Şekil 9.6.

Örneğin, Şekil 9.7'nin sağ tarafındaki Sipariş tablosuna bakarsanız, bir müşterinin müşteri olmak için sipariş vermesi gerekmediğini fark edeceksiniz. Başka bir deyişle, **çok tarafı** isteğe bağlıdır.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Cardinality-Optional-0-or-more-300x189.jpg)Şekil 9.7. A. Watt tarafından sıfırdan çoka isteğe bağlı ilişki sembolünün örnek kullanımı.

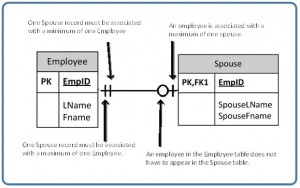
Şekil 9.7'deki ilişki sembolü aşağıdaki şekilde de okunabilir:

* Sol taraf: Sipariş varlığı, Müşteri tablosunda en az bir, en fazla bir ilişkili varlık içermelidir.
* Sağ taraf: Bir müşteri  en az sıfır sipariş veya en fazla çok sipariş  verebilir .

Şekil 9.8 sıfır ve bir ile başka bir tür isteğe bağlı ilişki sembolü gösterir, bu sıfır VEYA bir anlamına gelir.  Bir **taraf** isteğe bağlıdır.

[Ch-9-İsteğe bağlı-Ok-2](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-Optional-Arrow-2.jpg)Şekil 9.8.

Şekil 9.9 sıfırdan bire sembolünün nasıl kullanılabileceğine dair bir örnek göstermektedir.

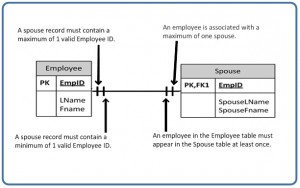
[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Cardinality-Optional-0-or-1-300x188.jpg)Şekil 9.9. A. Watt tarafından sıfırdan bire isteğe bağlı ilişki sembolünün örnek kullanımı.

Zorunlu ilişkiler

*Zorunlu* bir ilişkide , bir varlık oluşumu, karşılık gelen bir varlık oluşumunu gerektirir. Bu ilişki için sembol, Şekil 9.10'da gösterildiği gibi *yalnızca bir tane* gösterir . Bir taraf zorunludur.

[Bölüm-9-birdenbire-1](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-oneToOne-1.jpg)Şekil 9.10

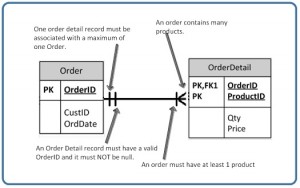
Tek ve zorunlu sembolün nasıl kullanıldığına dair bir örnek için Şekil 9.11'e bakın.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-Mandatory-Cardinality-one-to-one-300x188.jpg)Şekil 9.11. A. Watt'ın tek ve yalnızca bir zorunlu ilişki sembolüne ait örneği.

**Şekil 9.12, çok tarafın** zorunlu olduğu bir-çok ilişki sembolünün nasıl göründüğünü göstermektedir .

[Bölüm-9-birdençoğa-3](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-oneToMany-3.jpg)Şekil 9.12.

Birden çoğa sembolünün nasıl kullanılabileceğine dair bir örnek için Şekil 9.13'e bakın.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-Mandatory-Cardinality-one-to-Many-300x188.jpg)Şekil 9.13. A. Watt'ın bire çok zorunlu ilişki sembolüne bir örnek.

Şimdiye kadar bir ilişki sembolünün en iç tarafının  (Şekil 9.14'teki sembolün sol tarafında) olabileceğini gördük .0 (sıfır) kardinalite ve çok sayıda bağlantı (Şekil 9.14'teki sembolün sağ tarafında gösterilmiştir) veya bir (gösterilmemiştir).

[Ch-9-Sıfır-veya-Çok-1](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-Zero-or-Many-1.jpg)Şekil 9.14

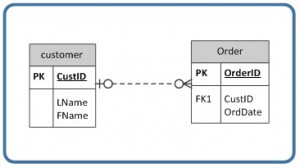
Ancak, Şekil 9.15'te gösterildiği gibi 0 (sıfır) bağlantıya sahip olamaz. Bağlantı yalnızca 1 olabilir.

[Ch-9-Sıfır-veya-Çok-2](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-Zero-or-Many-2.jpg)Şekil 9.15.

Bağlantı sembolleri maksimumları gösterir. Yani mantıksal olarak düşünürseniz, sol taraftaki bağlantı sembolü 0 (sıfır) gösteriyorsa, tablolar arasında hiçbir bağlantı olmazdı.

Şekil 9.16'daki gibi bir ilişki sembolünü okumanın yolu şu şekildedir.

* Order tablosundaki CustID değerinin Customer tablosunda da en az 0, en fazla 1 kez bulunması gerekmektedir.
* 0, Order tablosundaki CustID değerinin boş olabileceği anlamına gelir.
* En soldaki 1 (bağlantıyı temsil eden 0'dan hemen önce), Order tablosunda bir CustID varsa, bunun Customer tablosunda yalnızca bir kez bulunabileceğini belirtir.
* Kardinalite için 0 sembolünü gördüğünüzde iki şeyi varsayabilirsiniz: T
  1. Order tablosundaki FK boş değerlere izin verir ve
  2. FK, PK'nın bir parçası değildir çünkü PK'lar boş değerler içermemelidir.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-9-Cust-to-Order-ERD-300x168.jpg)Şekil 9.16. A. Watt'ın Müşteri tablosu ile Sipariş tablosu arasındaki ilişki.

Anahtar Terimler

**iş kuralları** : gereksinimleri toplarken kullanıcılardan elde edilir ve kardinaliteyi belirlemek için kullanılır

**kardinalite** : ilgili bir varlığın bir oluşumuyla ilişkili varlık oluşumlarının minimum ve maksimum sayısını ifade eder

**bağlantı** : iki tablo arasındaki ilişki, örneğin bire bir veya birden çoğa

**kısıtlamalar** : DBMS'lerin verilerin semantiği karşıladığını kontrol etmesini zorlayan kurallar

**varlık bütünlüğü** : her tablonun birincil bir anahtara sahip olmasını gerektirir; birincil anahtar veya herhangi bir parçası boş değerler içeremez

**tanımlayıcı ilişki** : birincil anahtar yabancı anahtarı içerir; ERD'de düz bir çizgiyle gösterilir

**bütünlük kısıtlamaları** : hangi veri değerlerinin izin verilip verilmediğini ve bir öznitelik için hangi formatın uygun olduğunu belirten mantıksal ifadeler

**zorunlu ilişki** : bir varlığın varlığı, karşılık gelen bir varlığın varlığını gerektirir.

**tanımlayıcı olmayan ilişki** : birincil anahtarda yabancı anahtarı içermez; ERD'de noktalı çizgiyle gösterilir

**isteğe bağlı ilişki** :  FKoluşumuna sahip olması gerekmez

**yetim kayıt** : yabancı anahtar değeri, birincil anahtarın bulunduğu varlık olan ilgili varlıkta bulunmayan kayıt

**referans bütünlüğü** : yabancı anahtarın eşleşen bir birincil anahtara sahip olması veya boş olması gerekir

**ilişkisel veritabanı yönetim sistemi (RDBMS)** : IBM'in San Jose Araştırma Laboratuvarı'ndan EF Codd tarafından tanıtılan ilişkisel modele dayalı popüler bir veritabanı sistemi

**ilişki türü :** ERD'deki  iki tablo arasındaki ilişkinin türü (tanımlayıcı veya tanımlayıcı olmayan); bu ilişki iki tablo arasına çizilen bir çizgiyle gösterilir.

**ER Modelleme**

Varlık ilişkisel (ER) modeli için geliştirilen önemli bir teori, işlevsel bağımlılık (FD) kavramını içerir. Bunu incelemenin amacı, veriler arasındaki ilişkilere ilişkin anlayışınızı geliştirmek ve pratik veritabanı tasarımına yardımcı olmak için yeterli biçimsellik kazanmaktır.

Kısıtlamalar gibi, FD'ler de uygulama alanının semantiğinden çizilir. Esasen,   *işlevsel bağımlılıklar* bireysel niteliklerin nasıl ilişkili olduğunu açıklar. FD'ler bir ilişki içindeki nitelikler arasında bir tür kısıtlamadır ve iyi bir ilişkisel şema tasarımına katkıda bulunur. Bu bölümde şunlara bakacağız:

* Fonksiyonel bağımlılığın temel teorisi ve tanımı
* Şema tasarımlarını iyileştirme metodolojisi , normalizasyon olarak da adlandırılır

**İlişkisel Tasarım ve Yedeklilik**

Genel olarak, iyi bir ilişkisel veritabanı tasarımı gerekli tüm nitelikleri ve ilişkileri yakalamalıdır . Tasarım bunu minimum miktarda depolanmış bilgi ve gereksiz veri olmadan yapmalıdır .

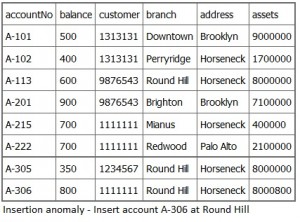
Veritabanı tasarımında, yedeklilik genellikle istenmeyen bir durumdur çünkü güncellemelerden sonra tutarlılığı korumada sorunlara neden olur. Ancak, yedeklilik bazen performans iyileştirmelerine yol açabilir ; örneğin, yedeklilik verileri bağlamak için bir *birleştirme* yerine kullanılabildiğinde . Bir *birleştirme,* iki ilişkili tabloya dayalı olarak bilgi edinmeniz gerektiğinde kullanılır.

Şekil 10.1'i düşünün: müşteri 1313131 iki kez görüntülenir, bir kez hesap no. A-101 için ve bir kez daha hesap no. A-102 için. Bu durumda, tabloda silme anormallikleri olmasına rağmen müşteri numarası gereksiz değildir. Ayrı bir müşteri tablosu olması bu sorunu çözer. Ancak, bir şube adresi değişirse, birden fazla yerde güncellenmesi gerekir. Müşteri numarası olduğu gibi tabloda bırakılırsa, şube tablosuna ihtiyacınız olmaz ve birleştirme gerekmez ve performans iyileştirilir.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Bank-Accounts-1-300x197.jpg)Şekil 10.1. Banka hesapları ve şubelerde kullanılan yedekliliğe bir örnek.

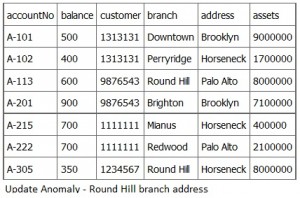
**Ekleme Anomalisi**

Bir tabloya tutarsız bilgiler eklediğinizde  bir *ekleme anormalliği* oluşur. Şekil 10.2'deki A-306 hesap numarası gibi yeni bir kayıt eklediğimizde , şube verilerinin mevcut satırlarla tutarlı olduğunu kontrol etmemiz gerekir.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Insertion-Anomaly-Banking-Accounts-300x222.jpg)Şekil 10.2. Bir ekleme anomalisinin örneği.

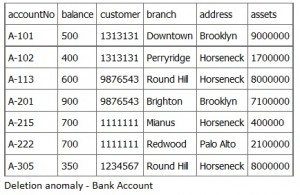
**Güncelleme Anomalisi**

Şekil 10.3'teki Round Hill şubesi gibi bir şubenin adresi değişirse, o şubeye atıfta bulunan tüm satırları güncellememiz gerekir. Mevcut bilgileri yanlış bir şekilde değiştirmeye *güncelleme anomalisi* denir .

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Update-Anomaly-Bank-Accounts-300x198.jpg)Şekil 10.3. Güncelleme anomalisine örnek.

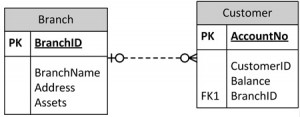
**Silme Anomalisi**

 Silinmemesi gereken öznitelikler içerebilecek bir kaydı sildiğinizde  bir *silme anormalliği* oluşur. Örneğin, Şekil 10.4'teki Downtown şubesindeki A-101 hesabı gibi bir şubedeki son hesapla ilgili bilgileri kaldırırsak, şube bilgilerinin tümü kaybolur.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Deletion-anomaly-Bank-Account-300x195.jpg)Şekil 10.4. Silme anomalisine örnek.

A-101 satırını silmenin sorunu, Downtown şubesinin nerede olduğunu bilmememiz ve 1313131 numaralı müşteriye ait tüm bilgileri kaybetmemizdir. Bu tür güncelleme veya silme sorunlarından kaçınmak için, orijinal tabloyu her tablonun diğer tablolarla minimum örtüşmeye sahip olduğu birkaç küçük tabloya ayırmamız gerekir.

Her banka hesap tablosu, Şekil 10.5'te gösterildiği gibi Şube veya Müşteri gibi yalnızca bir kuruluşa ait bilgileri içermelidir.

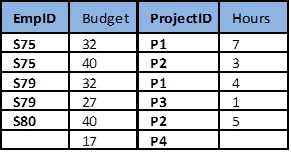
[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-10-Branch-to-Customer-ERD-300x117.jpg)Şekil 10.5. A. Watt'ın her biri bir varlık içeren banka hesap tablolarına ait örnekler.

Bu uygulamayı takip etmek, şube bilgisi eklendiğinde veya güncellendiğinde yalnızca bir kaydı etkileyeceğini garanti eder. Bu nedenle, müşteri bilgisi eklendiğinde veya silindiğinde, şube bilgisi yanlışlıkla değiştirilmeyecek veya yanlış kaydedilmeyecektir.

Örnek: çalışan proje tablosu ve anomaliler

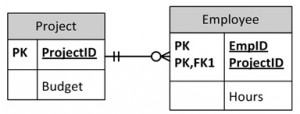
Şekil 10.6 bir çalışan proje tablosunun örneğini göstermektedir. Bu tablodan şunu varsayabiliriz :

1. EmpID ve ProjectID bileşik bir PK'dır.
2. Proje Kimliği Bütçeyi belirler (yani, Proje P1'in 32 saatlik bir bütçesi vardır).

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-10-ProjectEmp-table.jpg)Şekil 10.6. A. Watt tarafından hazırlanan bir çalışan proje tablosu örneği.

Şimdi, aşağıdaki adımlar sırasında bu tabloda oluşabilecek bazı olası anormalliklere bakalım .

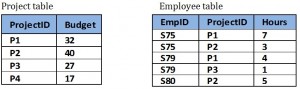
1. Eylem: {S85,35,P1,9} satırını ekle
2. Sorun: Çakışan bütçelere sahip iki tuple var
3. Eylem: {S79, 27, P3, 1} tuplesini sil
4. Sorun : Adım #3, P3 projesinin bütçesini siler
5. Eylem : {S75, 32, P1, 7} ikilisini {S75, 35, P1, 7} olarak güncelle
6. Sorun: Adım #5, P1 projesinin bütçesi için farklı değerlere sahip iki grup oluşturur
7. Çözüm: Şekil 10.7'de gösterildiği gibi Projeler ve Çalışanlar için ayrı birer tablo oluşturun.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-10-Project-to-Emp-ERD-300x114.jpg)Şekil 10.7. Çözüm: A. Watt'ın Proje ve Çalışan için ayrı tabloları.

**Anomalilerden Nasıl Kaçınılır**

Anomalisiz tablolar oluşturmanın en iyi yaklaşımı, tabloların normalleştirildiğinden emin olmaktır ve bu, işlevsel bağımlılıkları anlayarak gerçekleştirilir. FD, bir tablodaki tüm özniteliklerin o tabloya ait olduğundan emin olur. Başka bir deyişle, gereksizlikleri ve anomalileri ortadan kaldıracaktır.

Örnek: Proje ve Çalışan tablolarını ayırın

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-10-Project-and-Emp-tables-300x89.jpg)Şekil 10.8. A. Watt tarafından hazırlanan, verileri içeren Proje ve Çalışan tablolarını ayırın.

Verileri ayrı ayrı Proje ve Çalışan tablolarını kullanarak tutarak:

1. Bütçede değişiklik yapılması durumunda herhangi bir anormallik oluşmayacaktır.
2. Hiçbir çalışanın atanmadığı projeler için kukla değerlere gerek yoktur.
3. Bir çalışanın katkısı silindiğinde hiçbir önemli veri kaybolmaz.
4. Çalışanın katkısının eklenmesi durumunda herhangi bir anormallik oluşmaz.

Anahtar Terimler

**silme anomalisi** : silinmemesi gereken öznitelikleri içerebilecek bir kaydı sildiğinizde oluşur

**işlevsel bağımlılık (FD)** : bireysel niteliklerin nasıl ilişkili olduğunu açıklar

**ekleme anormalliği** : bir tabloya tutarsız bilgi eklediğinizde oluşur

**birleştirme** :  İki ilişkili tabloya dayalı bilgi edinmeniz gerektiğinde kullanılır

**güncelleme anormalliği** : mevcut bilgileri hatalı bir şekilde değiştirme

**Normalizasyon**

Normalizasyon, veritabanı tasarım sürecinin bir parçası olmalıdır. Ancak, normalizasyon sürecini ER modelleme sürecinden ayırmak zordur, bu nedenle iki teknik eş zamanlı olarak kullanılmalıdır.

Bir kuruluşun veri gereksinimleri ve operasyonlarının büyük resmini veya makro görünümünü sağlamak için bir varlık ilişkisi diyagramı (ERD) kullanın. Bu, ilgili varlıkları, niteliklerini ve ilişkilerini tanımlamayı içeren yinelemeli bir süreçle oluşturulur.

Normalizasyon prosedürü belirli varlıkların özelliklerine odaklanır ve ERD içindeki varlıkların mikro görünümünü temsil eder.

**Normalizasyon Nedir?**

*Normalizasyon,* tasarım içgörüleri sağlayan ilişkisel teorinin bir dalıdır. Bir tabloda ne kadar yedeklilik olduğunu belirleme sürecidir. Normalizasyonun hedefleri şunlardır:

* İlişkisel bir şema içindeki yedeklilik düzeyini karakterize edebilme
* Fazlalığı ortadan kaldırmak için şemaları dönüştürme mekanizmaları sağlayın

Normalizasyon teorisi, işlevsel bağımlılıklar teorisinden büyük ölçüde yararlanır. Normalizasyon teorisi altı normal form (NF) tanımlar. Her normal form, bir şemanın karşılaması gereken bir dizi bağımlılık özelliğini içerir ve her normal form, güncelleme anomalilerinin varlığı ve/veya yokluğu hakkında garantiler verir. Bu, daha yüksek normal formların daha az yedekliliğe sahip olduğu ve sonuç olarak daha az güncelleme sorunu olduğu anlamına gelir.

**Normal Formlar**

Herhangi bir veritabanındaki tüm tablolar, daha sonra ele alacağımız normal formlardan birinde olabilir. İdeal olarak, yalnızca PK'dan FK'ya kadar minimum yedeklilik isteriz. Diğer her şey diğer tablolardan türetilmelidir. Altı normal form vardır, ancak yalnızca ilk dördüne bakacağız:

* Birinci normal form (1NF)
* İkinci normal form (2NF)
* Üçüncü normal form (3NF)
* Boyce-Codd normal formu (BCNF)

BCNF nadiren kullanılır.

**Birinci Normal Form (1NF)**

*Birinci normal formda* , her satır ve sütunun kesişiminde yalnızca tek değerlere izin verilir; dolayısıyla tekrar eden gruplar yoktur.

Tekrarlayan bir grup içeren bir ilişkiyi normalleştirmek için tekrarlayan grubu kaldırın ve iki yeni ilişki oluşturun.

Yeni ilişkinin PK'sı, özgün tanımlama için orijinal ilişkinin PK'sı ile yeni oluşturulan ilişkiden alınan bir niteliğin birleşimidir.

1NF için süreç

1NF sürecini açıklamak için örnek olarak, bir Okul veritabanından alınan **aşağıdaki** Student\_Grade\_Report tablosunu **kullanacağız** .

**Öğrenci\_Not\_Raporu** (ÖğrenciNo, ÖğrenciAdı, Anadal, DersNo, DersAdı, EğitmenNo, EğitmenAdı, EğitmenYeri, Not)

* Öğrenci Not Raporu tablosunda tekrar eden grup ders bilgisidir. Bir öğrenci birçok ders alabilir.
* Tekrar eden grubu kaldırın. Bu durumda, her öğrencinin ders bilgisidir.
* Yeni masanız için PK'yı belirleyin.
* PK, nitelik değerini (ÖğrenciNo ve DersNo) benzersiz bir şekilde tanımlamalıdır.
* **Ders ve öğrenciye ait tüm öznitelikler kaldırıldığında geriye öğrenci ders tablosu ( StudentCourse** ) kalır .
* Öğrenci tablosu ( **Öğrenci** ) artık tekrar eden grubun kaldırıldığı birinci normal formdadır.
* Aşağıda iki yeni tablo gösterilmektedir.

**Öğrenci** ( ÖğrenciNo , ÖğrenciAdı, Anadal)

**ÖğrenciDers** ( ÖğrenciNo, DersNo , DersAdı, EğitmenNo, EğitmenAdı, EğitmenKonumu, Sınıf)

1NF anomalileri nasıl güncellenir

**ÖğrenciDers** ( ÖğrenciNo, DersNo , DersAdı, EğitmenNo, EğitmenAdı, EğitmenKonumu, Sınıf)

* Yeni bir ders eklemek için bir öğrenciye ihtiyacımız var.
* Ders bilgilerinin güncellenmesi gerektiğinde tutarsızlıklar olabilir.
* *Bir öğrenciyi* silmek için , aynı zamanda bir derse ait kritik bilgileri de silebiliriz.

**İkinci Normal Form (2NF)**

*İkinci normal form* için , ilişki ilk önce 1NF'de olmalıdır. İlişki, yalnızca ve yalnızca PK tek bir niteliği içeriyorsa otomatik olarak 2NF'dedir.

Eğer ilişkinin bileşik bir PK'si varsa, o zaman anahtar olmayan her bir nitelik PK'nin tamamına tamamen bağımlı olmalı ve PK'nin bir alt kümesine bağımlı olmamalıdır (yani kısmi bağımlılık veya artırma olmamalıdır ).

2NF için süreç

2NF'ye geçebilmek için masanın öncelikle 1NF'de olması gerekir.

* Öğrenci tablosu tek sütunlu PK'ya sahip olduğundan zaten 2NF'dedir.
* Öğrenci Ders tablosunu incelediğimizde, tüm niteliklerin PK'ya tamamen bağımlı olmadığını görüyoruz; özellikle, tüm ders bilgileri. Tamamen bağımlı olan tek nitelik nottur.
* Ders bilgilerinin yer aldığı yeni tabloyu tanımlayın.
* Yeni tablonun PK'sını tanımlayın.
* Aşağıda üç yeni tablo gösterilmektedir.

**Öğrenci** ( ÖğrenciNo , ÖğrenciAdı, Anadal)

**Ders Notu** ( ÖğrenciNo, DersNo , Not)

**DersEğitmeni** ( DersNo , DersAdı, EğitmenNo, EğitmenAdı, EğitmenKonumu)

2NF anomalileri nasıl güncellenir

* Yeni bir eğitmen eklerken bir kursa ihtiyacımız var.
* Ders bilgilerinin güncellenmesi eğitmen bilgilerinde tutarsızlıklara yol açabilir.
* Bir dersi sildiğinizde eğitmen bilgileri de silinebilir.

**Üçüncü Normal Form (3NF)**

*Üçüncü normal formda* olmak için , ilişki ikinci normal formda olmalıdır. Ayrıca tüm geçişli bağımlılıklar kaldırılmalıdır; anahtar olmayan bir öznitelik, başka bir anahtar olmayan özniteliğe işlevsel olarak bağımlı olmayabilir.

3NF için süreç

* Geçişli ilişkiye sahip her tablodan geçişli ilişkideki tüm bağımlı nitelikleri ortadan kaldırın.
* Bağımlılıkları kaldırarak yeni tablo(lar) oluşturun.
* Her tablonun bir belirleyiciye sahip olduğundan ve hiçbir tablonun uygunsuz bağımlılıklar içermediğinden emin olmak için yeni tablo(lar)ı ve değiştirilen tablo(lar)ı kontrol edin.
* Aşağıdaki dört yeni tabloyu inceleyin.

**Öğrenci** (ÖğrenciNo, ÖğrenciAdı, Anadal)

**Ders Notu** ( ÖğrenciNo, DersNo , Not)

**Ders** ( DersNo , DersAdı, EğitmenNo)

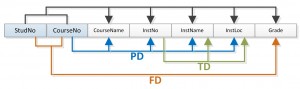
**Eğitmen** (EğitmenNo, EğitmenAdı, EğitmenKonumu)

Bu aşamada, üçüncü normal formda herhangi bir anormallik olmamalıdır . Bu örnek için bağımlılık diyagramına (Şekil 12.1) bakalım. İlk adım, yukarıda tartışıldığı gibi tekrar eden grupları kaldırmaktır.

**Öğrenci** (ÖğrenciNo, ÖğrenciAdı, Anadal)

**ÖğrenciDers** (ÖğrenciNo, DersNo, DersAdı, EğitmenNo, EğitmenAdı, EğitmenKonumu, Sınıf)

Okul veritabanı için normalleştirme sürecini özetlemek için Şekil 12.1'de gösterilen bağımlılıkları inceleyin.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/Ch-11-Dependency-Diagram-School-300x89.jpg)Şekil 12.1 Bağımlılık diyagramı, A. Watt.

Şekil 12.1’de kullanılan kısaltmalar şu şekildedir:

* PD: kısmi bağımlılık
* TD: geçişli bağımlılık
* **FD: tam bağımlılık (Not: FD genellikle işlevsel** bağımlılık anlamına gelir . FD'yi tam bağımlılık için bir kısaltma olarak kullanmak yalnızca Şekil 12.1'de kullanılmıştır.)

**Boyce-Codd Normal Formu (BCNF)**

Bir tablonun birden fazla aday anahtarı olduğunda, ilişki 3NF'de olsa bile anomaliler ortaya çıkabilir. *Boyce-Codd normal formu* 3NF'nin özel bir durumudur. Bir ilişki, yalnızca ve yalnızca her belirleyici bir aday anahtarsa ​​BCNF'dedir.

BCNF Örneği 1

Aşağıdaki tabloyu ( **St\_Maj\_Adv** ) ele alalım.

|  |  |  |
| --- | --- | --- |
| **Öğrenci\_kimliği** | **Ana** | **Danışman** |
| 111 | Fizik | Demirci |
| 111 | Müzik | Şan |
| 320 | Matematik | Dobbs |
| 671 | Fizik | Beyaz |
| 803 | Fizik | Demirci |

Bu tablo için semantik *kurallar* ( veritabanına uygulanan iş kuralları) şunlardır :

1. Her Öğrenci birden fazla alanda uzmanlaşabilir.
2. Her Anadal için, bir Öğrencinin yalnızca bir Danışmanı vardır.
3. Her anadalın birden fazla danışmanı vardır.
4. Her Danışman sadece bir Binbaşıya danışmanlık yapar .
5. Her Danışman, bir Anadalda birden fazla Öğrenciye danışmanlık yapar .

Bu tablonun işlevsel bağımlılıkları aşağıda listelenmiştir. İlki aday anahtardır; ikincisi değildir.

1. Öğrenci\_kimliği, Anadal ——> Danışman
2. Danışman ——> Binbaşı

Bu tabloya ilişkin anormallikler şunlardır:

1. Sil – öğrenci danışman bilgilerini siler
2. Ekle – yeni bir danışmanın bir öğrenciye ihtiyacı var
3. Güncelleme – tutarsızlıklar

**Not** : Hiçbir nitelik tek başına aday anahtar değildir.

PK Öğrenci\_kimliği, Bölüm veya Öğrenci\_kimliği, Danışman olabilir **.**

**St\_Maj\_Adv** ilişkisini BCNF'ye indirgemek için iki yeni tablo oluşturursunuz:

1. **St\_Adv** ( Öğrenci\_kimliği, Danışman )
2. **Adv\_Maj** ( Danışman , Binbaşı)

**St\_Adv** tablosu    

|  |  |
| --- | --- |
| **Öğrenci\_kimliği** | **Danışman** |
| 111 | Demirci |
| 111 | Şan |
| 320 | Dobbs |
| 671 | Beyaz |
| 803 | Demirci |

**Adv\_Maj** tablosu

|  |  |
| --- | --- |
| **Danışman** | **Ana** |
| Demirci | Fizik |
| Şan | Müzik |
| Dobbs | Matematik |
| Beyaz | Fizik |

BCNF Örneği 2

Aşağıdaki tabloyu ( **Client\_Interview )** ele alalım .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MüşteriNo** | **RöportajTarihi** | **RöportajZamanı** | **PersonelHayır** | **OdaHayır** |
| CR76 | 13-May-02 | 10.30 | SG5 | G101 |
| CR56 | 13-May-02 | 12.00 | SG5 | G101 |
| CR74 | 13-May-02 | 12.00 | SG37 | G102 |
| CR56 | 1-Temmuz-02 | 10.30 | SG5 | G102 |

FD1 – Müşteri No, Görüşme Tarihi –> Görüşme Süresi, Personel No, Oda No (PK)

FD2 – personelNo, röportajTarihi, röportajSaati –> istemciNO (aday anahtarı: CK)

FD3 – odaNo, röportajTarihi, röportajZamanı –> personelNo, müşteriNo (CK)

FD4 – personelNo, röportajTarihi –> odaNo

Bir ilişki, yalnızca ve yalnızca her belirleyici bir aday anahtarsa ​​BCNF'dedir. İlk üç FD'yi ( **Client\_Interview2** tablosu) ve  dördüncü FD için başka bir tabloyu ( **PersonelOdası tablosu) içeren bir tablo oluşturmamız gerekir.**

**Client\_Interview2** tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **MüşteriNo** | **RöportajTarihi** | **RöportajZamanı** | **PersonelHayır** |
| CR76 | 13-May-02 | 10.30 | SG5 |
| CR56 | 13-May-02 | 12.00 | SG5 |
| CR74 | 13-May-02 | 12.00 | SG37 |
| CR56 | 1-Temmuz-02 | 10.30 | SG5 |

**Personel Odası** masası

|  |  |  |
| --- | --- | --- |
| **PersonelHayır** | **RöportajTarihi** | **OdaHayır** |
| SG5 | 13-May-02 | G101 |
| SG37 | 13-May-02 | G102 |
| SG5 | 1-Temmuz-02 | G102 |

**Normalizasyon ve Veritabanı Tasarımı**

Veritabanı tasarımının normalleştirme süreci sırasında, tablo yapıları oluşturulmadan önce önerilen varlıkların gerekli normal formu karşıladığından emin olun. Birçok gerçek dünya veritabanı, zaman içinde uygunsuz şekilde değiştirilirse uygunsuz şekilde tasarlanmış veya anormalliklerle yüklenmiştir. Mevcut veritabanlarını yeniden tasarlamanız ve değiştirmeniz istenebilir. Tablolar uygun şekilde normalleştirilmezse bu büyük bir girişim olabilir.

Anahtar Terimler ve Kısaltmalar

**Boyce-Codd normal formu (BCNF)** : 3. NF'nin özel bir durumu

**birinci normal form (1NF):** her satır ve sütunun kesişiminde yalnızca tek değerlere izin verilir, bu nedenle tekrar eden gruplar yoktur

**normalizasyon** : bir tabloda ne kadar yedeklilik olduğunu belirleme süreci

**ikinci normal form (2NF)** : ilişki 1NF'de olmalı ve PK tek bir niteliği içermelidir

**anlamsal kurallar** : veritabanına uygulanan iş kuralları

**üçüncü normal form (3NF)** : ilişki 2NF'de olmalı ve tüm geçişli bağımlılıklar kaldırılmalıdır; anahtar olmayan bir nitelik, başka bir anahtar olmayan niteliğe işlevsel olarak bağımlı olmayabilir