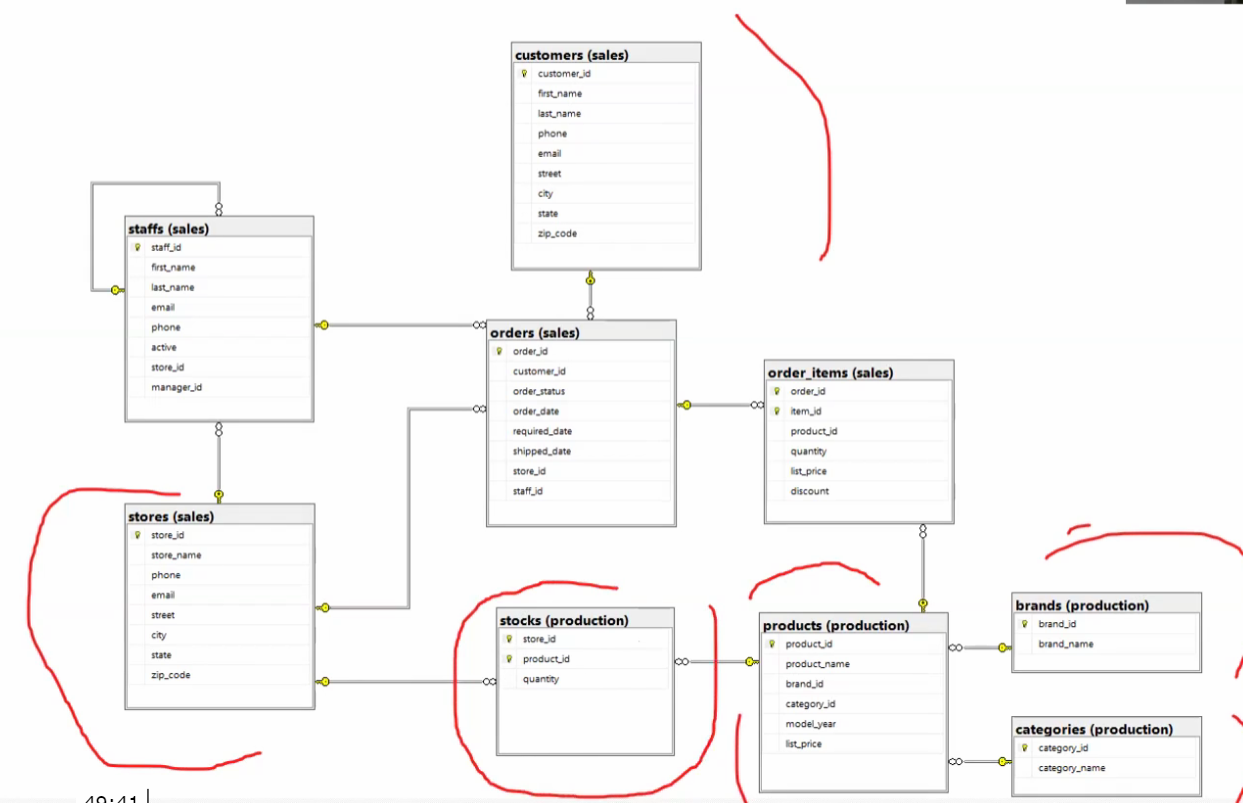
**DBMD\_INCLASS - 14 JULY 2021**

(PRIMARY KEY, COMPOSIT KEY, ANOMALI, FUNCTIONAL DEPENDENCY, PARTIAL DEPENDENCY, TRANSITIVE DEPENDENCY, NORMALİZASYON)



**COMPOSIT KEY**

Products entitisinin tablosu ile stores entitisinin tablosu arasında stocks relation’ı var.

Dikkat et, product tablosunda bir tane anahtar varken bu relation tablosunda (bridge tablosu) iki tane anahtar görüyorsunuz. Bu iki anahtarın biri ilişkinin bir tarafından gelen tablodan geliyor diğeri de ilişkinin diğer tarafındaki tablodan geliyor.

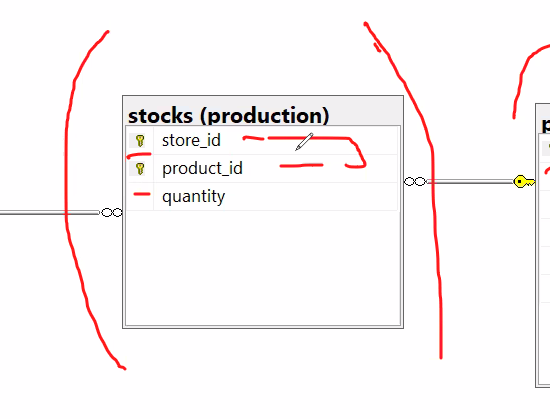
Stocks tablosunda store\_id tek başına bu tabloyu identify edemiyor. Product\_id de tek başına identify edemiyor. **Ancak ikisi beraber identify edebiliyor**.

Ancak beraberce o tabloyu identify edebilen bu iki primary key’e **composit key** diyoruz.

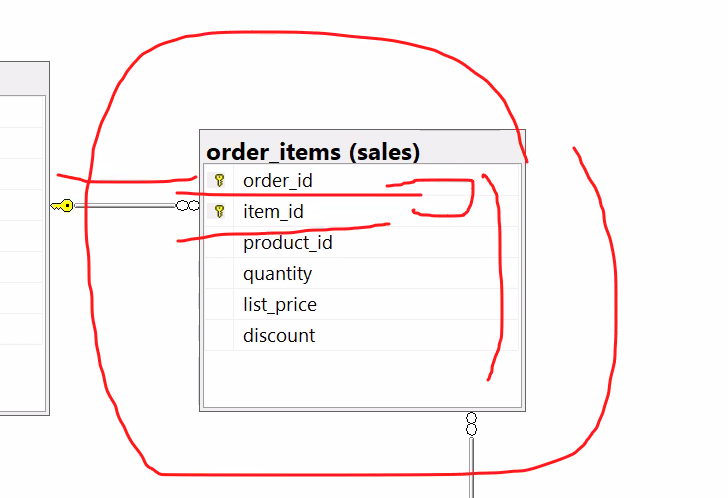
Order\_items tablosu da bir **relation tablosu** (orders) ile bir **entiti tablosunu** (products) birbirine bağlıyor. Dolayısıyla bir **bridge** oluyor.

Bridge tablolarındaki keyler relation tablolarından geldiği için bu keyler aynı zamanda **FOREIGN key** oluyor.

Yine buradaki order\_id ile item\_id keyleri beraber composit keyi oluşturuyor. Çünkü order\_id tek başına order\_items tablosunu identify edemiyor. Oysa orders tablosuna baktığımızda order\_id key’i o tablodaki tüm sütunları identiy edebiliyor. **Yani order\_id, orders tablosunda primary key olurken order\_items tablosunda composit key’in bir alt kümesi oluyor**.

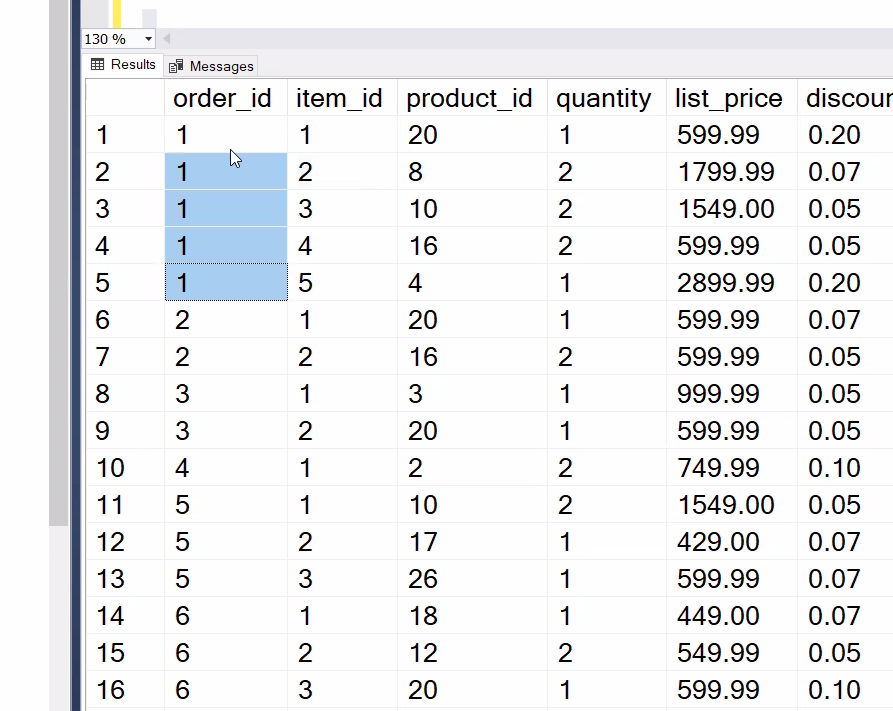


**Burada product\_id tek başına primary key olamıyor ancak store\_id ile birlikte olabiliyor**. O yüzden başka bir tablodan gelen store\_id ile yine başka bir tablodan gelen product\_id i beraber **composite key** yaptık. **İki sütun birden primary key oldu.**



Tek başına order\_id, order\_items tablosunun tüm alanlarını identify edemiyor. Bu yüzden item\_id ile birlikte primary key yapıyoruz.

---------------------------------------------------------------------------------------------------

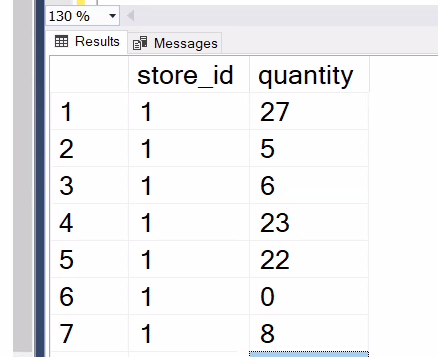
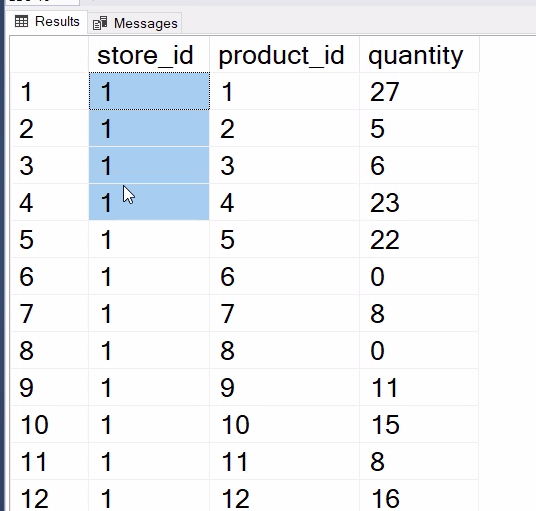


**PRIMARY KEY: bir tabloyu identify etme yeteneğine sahip keydir.**

Ama örneğin burada bakın (order\_id’lerden) 1 çoklanmış. 2. satırda farklı bir item var. Çünkü her farklı product için farklı bir item\_id var. O product’ın farklı bir order\_id’si yok, o da diğer 4 ürünle birlikte aynı order\_id’de sipariş edilmiş, dolayısıyla aynı order\_id’lerine sahip olduğundan ona farklı bir order\_id verilemiyor.

3, 4 ve 5. satırda da aynı durum söz konusu. Bu yüzden order\_id ve item\_id birlikte kullanıyoruz ve bunlar beraber bir primary key oluşturuyor. (2’si beraber olunca benzersiz oluyor)

----------------------------------------------------------------------------------------------



Bu tabloda hangi mağazada hangi üründen kaçar tane var bilgileri mevcut. o yüzden store id bilgisi olacak, product id olacak ve ürün sayısı bilgisi olacak.

Burada product\_id sütunu olmasaydı bu tablo bize bir şey ifade eder miydi? Hayır, quantitiy değerlerinin karşılığı olarak hepsinde 1 çoklamış ve bu yüzden hiç bir şey ifade etmiyor. Çünkü quantitiy sütunundaki değerlerin neye ait olduğunu belirleyemiyor, adresini belirleyemiyor yani tanımlayamıyor. Demek ki Store\_id sütunu bu tabloyu tek başına bize anlatamıyor, tanımlayamıyor. **Burada başka bir feature daha olması lazım. O da product\_id. Dikkat edersen her bir quantity değerine karşılık onu tanımlayan ayrı bir product\_id var.**

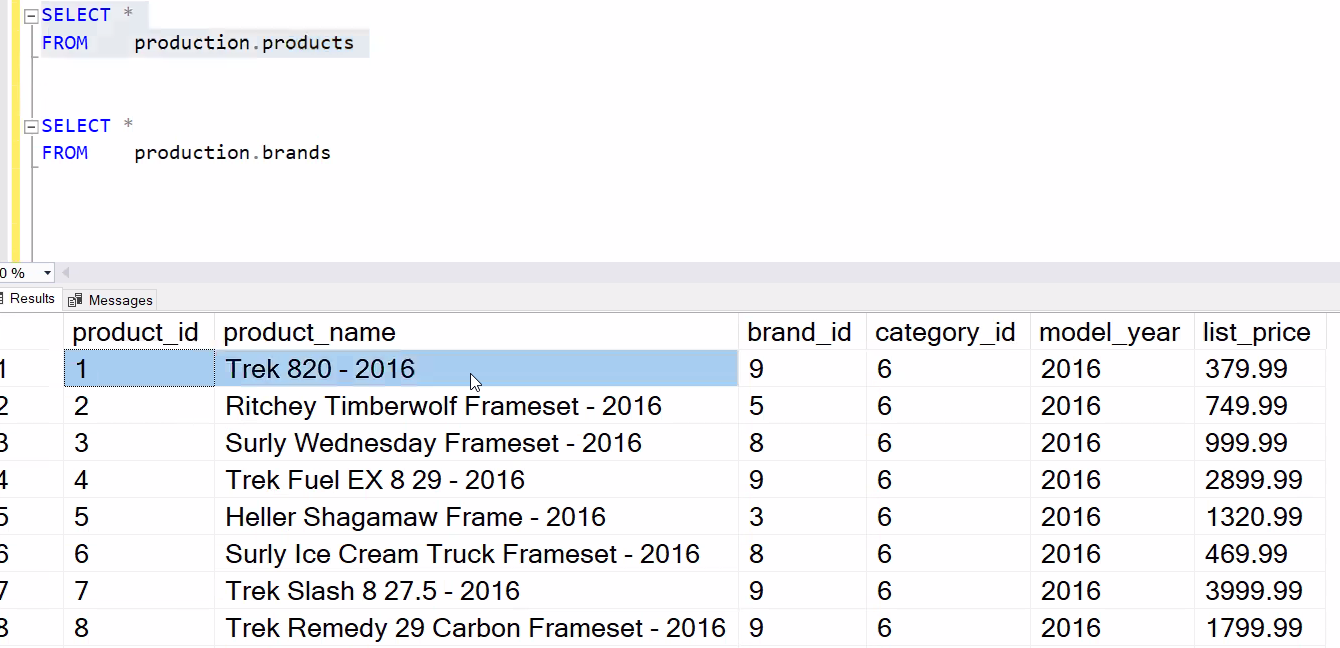
Peki bu tabloda brand\_name yok. Neden yok? Çünkü eğer her sütünü bir tabloda toplasaydı; mesela brand\_name’ler her bir product için çoklayacaktı, her bir product için tekrar tekrar yazılmasına neden olacaktı, tablo şişecekti, gereksiz yer kaplayacaktı, işlem esnasında sistemi daha fazla yoracaktı. Biz bu duruma **ANOMALİ** diyoruz.

Örneğin brand\_name dahil tüm sütunlar bir tek tabloda toplanmış olsaydı: diyelim yeni bir ürün oluştu ve mağazaya mevcut markalardan başka bir marka geldi. Bu durumda brand\_name sütununa yeni markayı eklediğimde onun karşılığına gelen product\_name’ler vs.ler hep NULL NULL değerler verecek ve boş alanlar oluşacaktı. Oysa brand name sütunu ayrı bir tabloda tutulsa o tabloya basit bir ilave olacak ve ihtiyacım olduğunda bu tabloya danışarak veya join ederek diğer bilgilere rahatlıkla ulaşabileceğim.

**ANOMALİ** :

Bütün bilgiler tek bir tabloda olursa bir bilgiyi silmek, değiştirmek veya update etmek istediğinde sorun yaşarsın. Yanlış bilgiyi silebilirsin ve bir entitiyi tamamen silebilirsin. Bunlara dikkat etmek gerekiyor. Bunun önüne geçmek için geliştirile geliştirile relational database kavramı oluşmuş ve tüm sütunların bir tabloda toplanması **anomali** olarak görülmüş ve bundan vazgeçilmiş. Anomalilerin giderilmesi işlemine de **normalizasyon** denilmiştir.

--------------------------------------------------------------------------------------

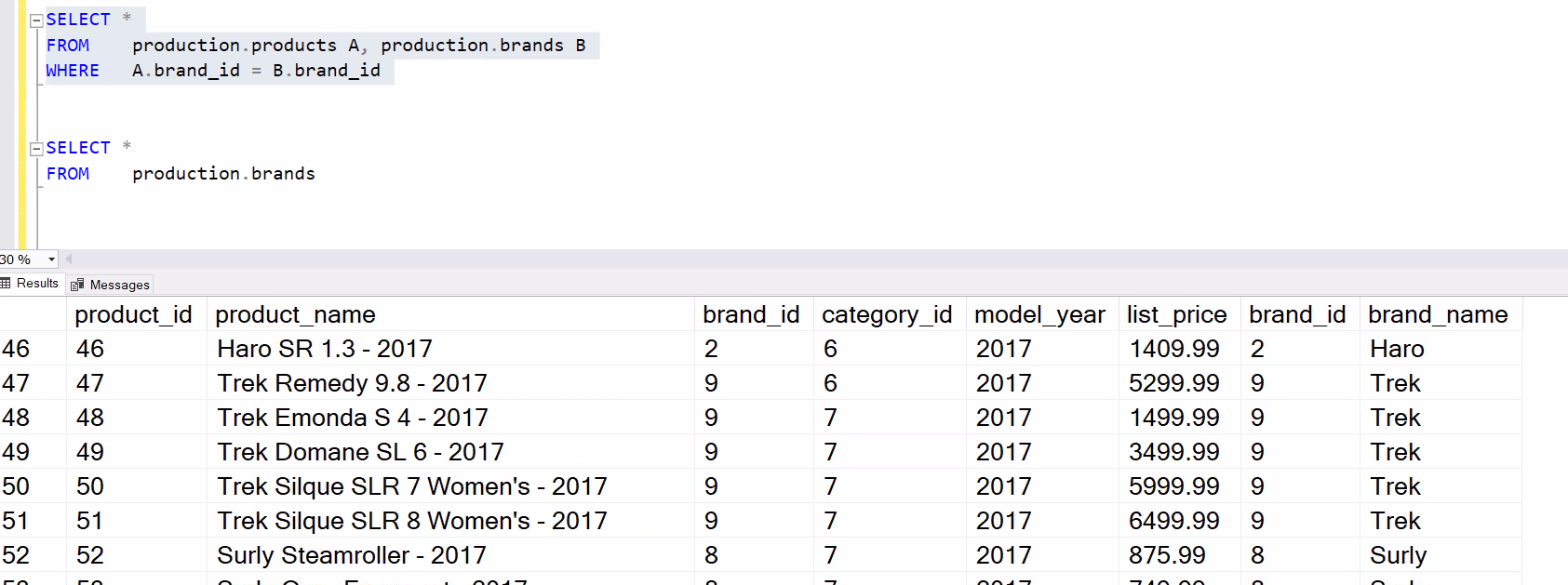


**Functional dependent** olması için o sütunun her bir değerinin **dependent** olduğu sütundaki **(ki bu primary keydir)** yalnız bir unique değer ile tanımlanabilmesi lazım. (*Aşağıda bu konu anlatılmıştır*)

Buradaki örnekte; Product\_name’i Trek 820-2016 olan ürün product\_id’ye functional dependent. Fakat category\_id değerleri product\_id ye functional dependent değil çünkü birden çok “6” değeri farklı product\_id’lerde tanımlı.

brand\_id ile bu ürünün hangi brande ait olduğunu ve category\_id ile hangi category’ye ait olduğunu anlatıyor.

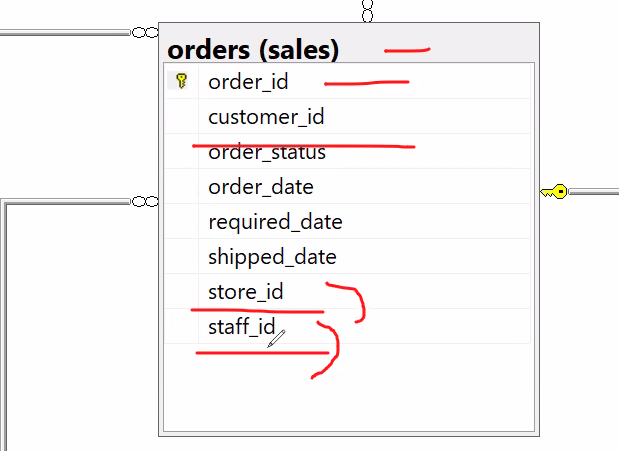
Bütün bilgileri product tablosunda tutsaydık çoklamalar olacaktı. Aşağıda gördüğün gibi brand\_name’ler çoklamış.



**Bu alanlar (sütunlar) çoğaldıkça tekrarlarla tablom şişecekti. Değişiklik yapmak istediğimde bu bana zorluklar çıkaracaktı. Bir tabloda bütün özelliklerin olması gereksiz olacaktır performansı da düşürecektir.**

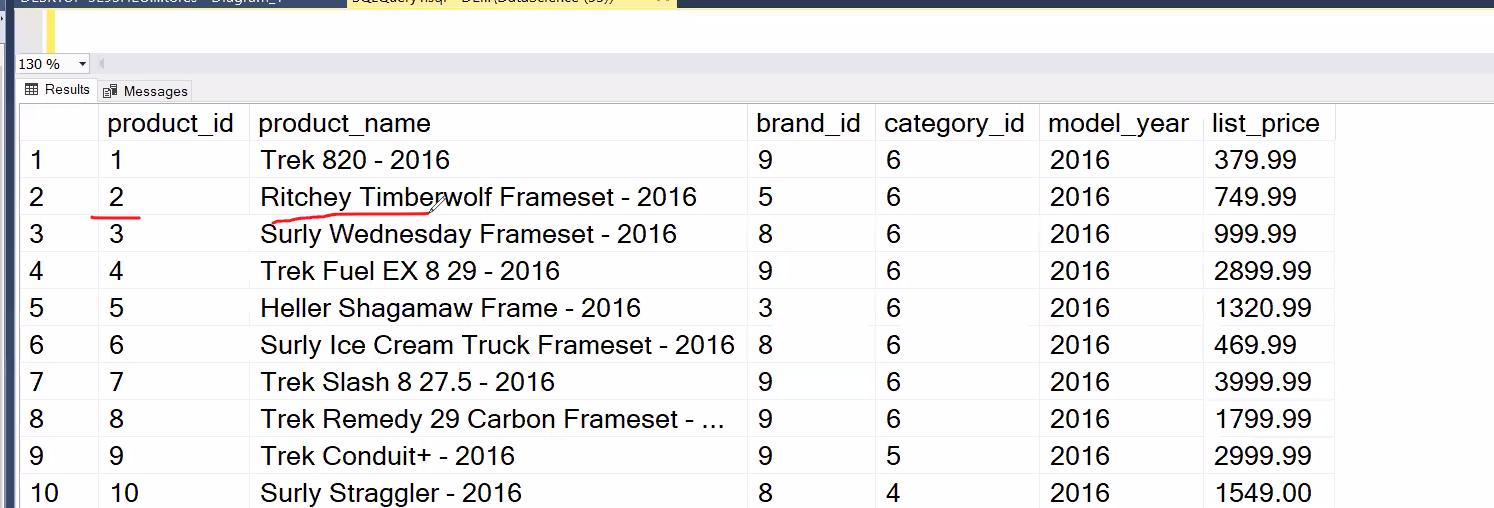
**İşte bunlara anomali deniyor**. Ve normalizasyonu bunun için yapıyoruz. mevcut olan bütün entitileri bölüyoruz, bilgileri atomize ediyoruz (parçalara ayırıyoruz)

Aşağıdaki tabloda store\_id ve staff\_id alanlarına gerek yok ama olmasının da zararı yok. Bunları asıl staffs.sales tablosunda tutuyor.

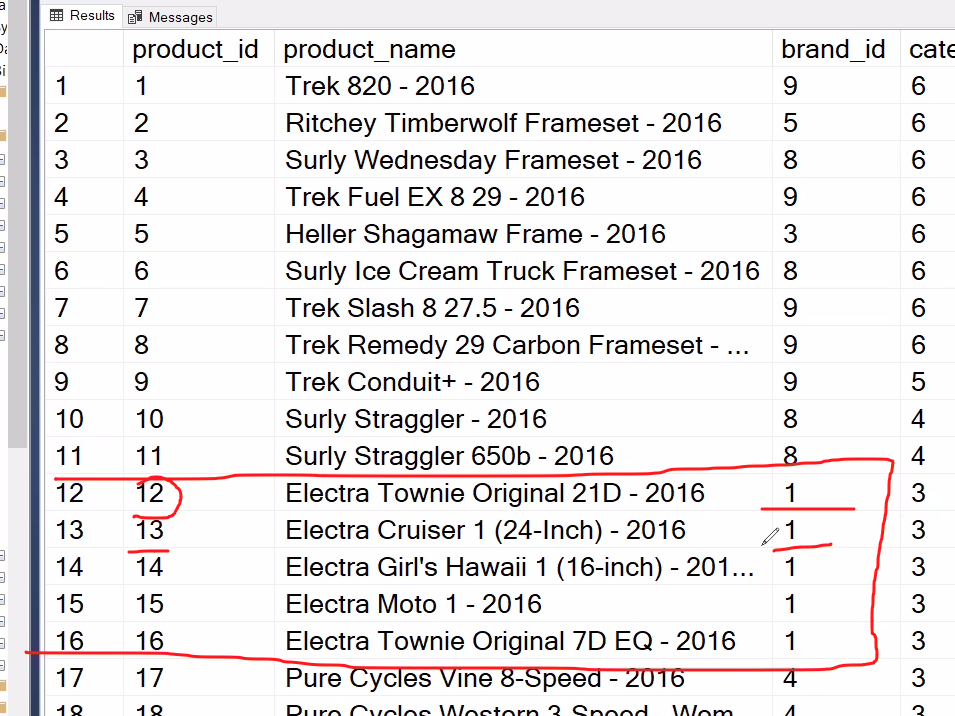


--------------------------------------------------------------------------------------

**FUNCTIONAL DEPENDENCY:**



Primary key bir tabloyu temsil ediyor, tam olarak tanımlıyor olmalı. Burada product\_name’lere baktığımızda Ritche Timberwolf.. ürünü yalnızca **bir tek** product\_id ile tanımlanıyor. Product\_name sütunu product\_id sütunu tarafından tanımlanabildiği (identify) için product\_name sütunu product\_id sütununa functional olarak bağlı. Buna **functional dependency** diyoruz.

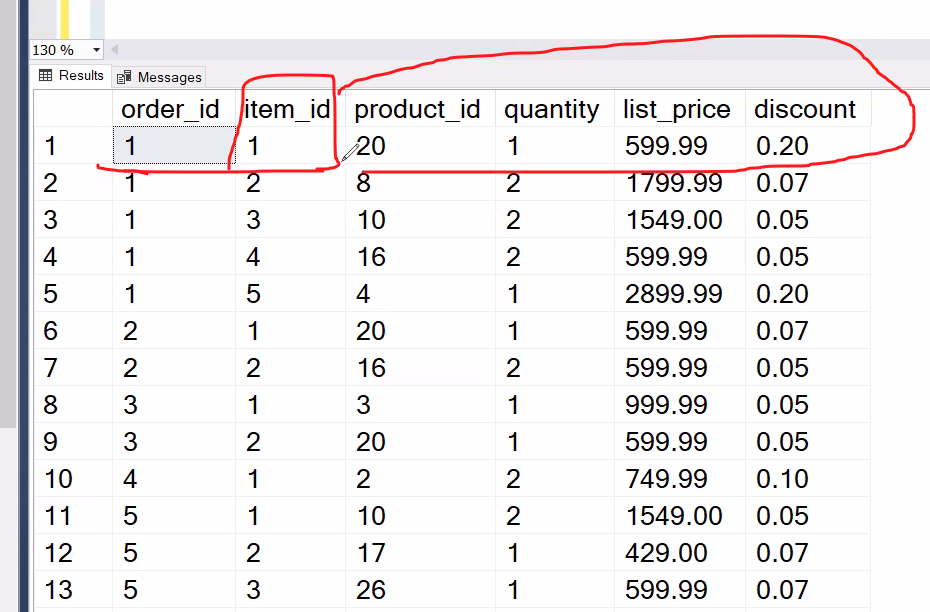


Peki product\_id, brand\_id yi identify ediyor mu? Hayır edemiyor. Bir product\_id, bir brand\_id yi getiremiyor. O zaman brand\_id fonksiyonel olarak product\_id’ye bağlı değil. 12 numaralı product\_id 1 numaralı brand\_id’ye karşılık gelirken 13 numara da yine 1 e karşılık geliyor.

**Bir sütunun başka bir sütunu identify etmesi demek, her bir değerine ait diğer sütunda tek bir değere işaret etmesi demek.** Buradaki örnekte 12 no lu product\_id sadece bir brand\_id’yi identify etmesi lazımken edemiyor. **Demek ki brand\_id, product\_id’ye functional dependent değil.**

---------------------------------------------------

**PARTIAL DEPENDENCY**



**Primary key e tüm sütunlar FULL FUNCTIONAL DEPENDENT olmalı**.

**Yani Primary key dışındaki sütunlar başka bir sütuna bağlı olamayacak (functional dependent olmayacak)**

Bu örnekte mesela item\_id ben tek başıma lists\_price sütununu temsil edeceğim diyemez. O zaman al o sütununu başka bir tablo oluşturun deriz. Order\_id de aynı şekilde tek başına bir sütunu tanımlıyorum diyemez. Burada list\_price’ı 599.99 olan birçok ürün var. ama herhangi bir 599.99 olan değerini yalnızca order\_id ve item\_id beraberce tanımlayabiliyor. Örneğin1. Satırdaki 599.99 fiyatı, order\_id’si 1 ve item\_id’si 1 olan ürüne ait. Fakat 4. Satırdaki 599.99 fiyatı, order\_id’si 1 ve item\_id’si 4 olan ürüne ait. (yani kompozit key)

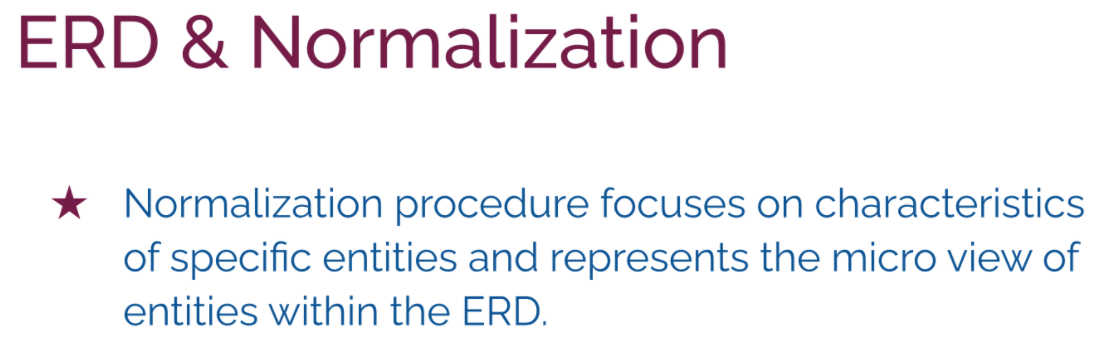
**Composit keyin kullanıldığı bir tabloda; eğer iki primary keyden biri, primary key olmayan bir sütunu identify edebiliyorsa buna PARTIAL DEPENDENCY deniyor**. Ve o iki sütun ayrılarak yeni bir tablo oluşturuluyor.

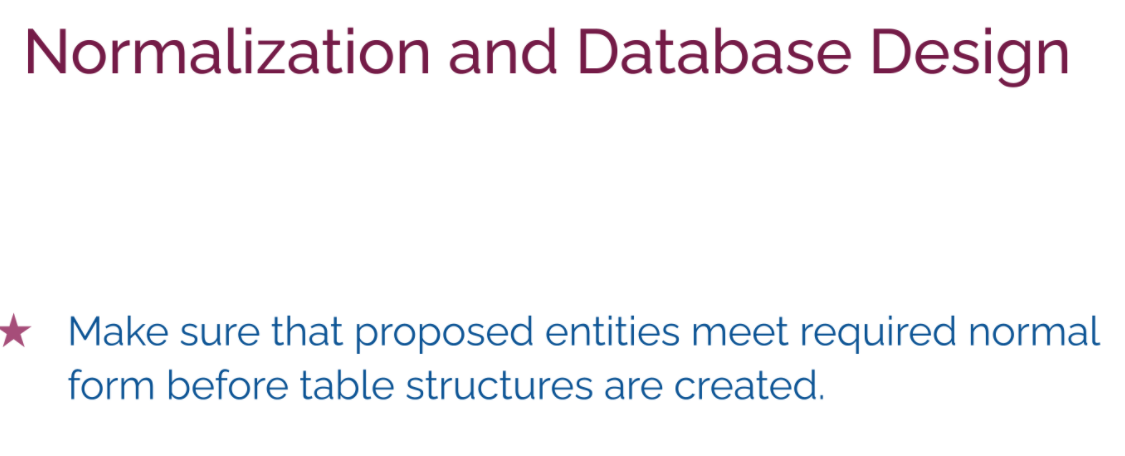
**TRANSITIVE DEPENDENCY :**

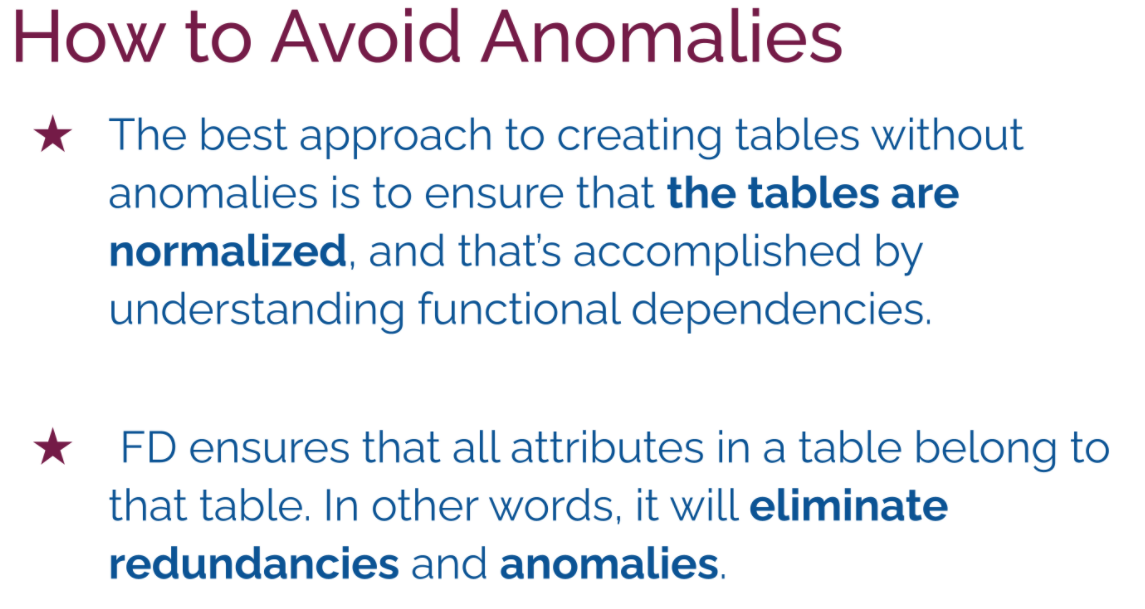
Eğer bir tabloda primary key olmayan (non-primary key) bir sütun başka bir primary key olmayan sütunu tanımlayabiliyorsa buna da **TRANSITIVE DEPENDENCY** deniyor. Ki bu da bir anomali olup normalize edilmesi gerek bir durumdur.

**--------------------------------------------------------------------------------------**

**NORMALIZATION:**

** **

****

****

**Entiti Relationship Diagram (ERD)**’da database’in tablolarının **normalize edilmiş halini** ve bunlar arasındaki ilişkileri gösteriyoruz. Eğer ralational database oluşturacaksak bu normalizasyon mutlaka yapılmalıdır.

Biz Bycstore’un ERD’sini gördük. **ERD neydi? normalize edilmiş bir database’in yapısını yansıtıyordu.** Relation’larını, entitilerini yansıtıyordu.

(Bütün bilgiler tek bir tabloda olursa bir bilgiyi silmek, değiştirmek veya update etmek istediğinde sorun yaşarsın. Yanlış bilgiyi silebilirsin ve bir entitiyi tamamen silebilirsin. Bunlara dikkat etmek gerekiyor. Bunun önüne geçmek için geliştirile geliştirile relational database kavramı oluşmuş ve tüm sütunların bir tabloda toplanması **anomali** olarak görülmüş ve bundan vazgeçilmiş. )

**Anomalilerin giderilmesi işlemine normalizasyon denilmektedir**.

Özetle normalizasyonun iki temel amacı vardır.

1. Veri tabanında veri tekrarlarını ortadan kaldırmak
2. Veri tutarlılığını (doğruluğunu) artırmak.

Yani:

Tablolar atomik bilgiler içerecek ve tekrar eden bilgi olmayacak

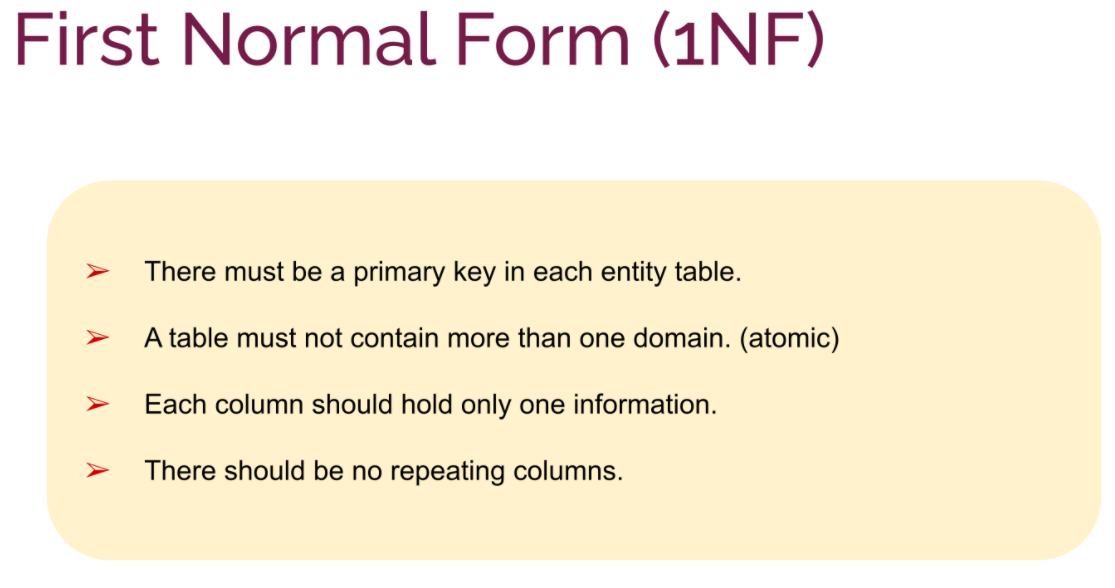
**Normal Forms:**

Basitçe tanımlamak gerekirse, normal formlar normalizasyon seviyeleridir. Bu seviyeler gereksiz veri tekrarlarını ne derecede engellediği ve tutarlılığı ne kadar sağladığına bağlı olarak derecelendirilir. Seviye yükseldikçe veri tutarlılığı artar, veri tekrarı düşer.

Normalizasyon seviyeleri 1NF (Birinci Normal Form), 2NF, 3NF, BCNF(Boyce-Codd Normal Form, 3.5NF'de denir), 4NF şeklinde adlandırılır ve yukarı doğru devam eder. Ancak daha yukarı normalizasyon seviyeleri çok nadiren kullanılır çünkü çoğu zaman uygulanması mümkün olmayabilir.

Bir database bunlardan ilk üçünü karşılıyorsa normalize edilmiş olarak kabul edilmiştir.

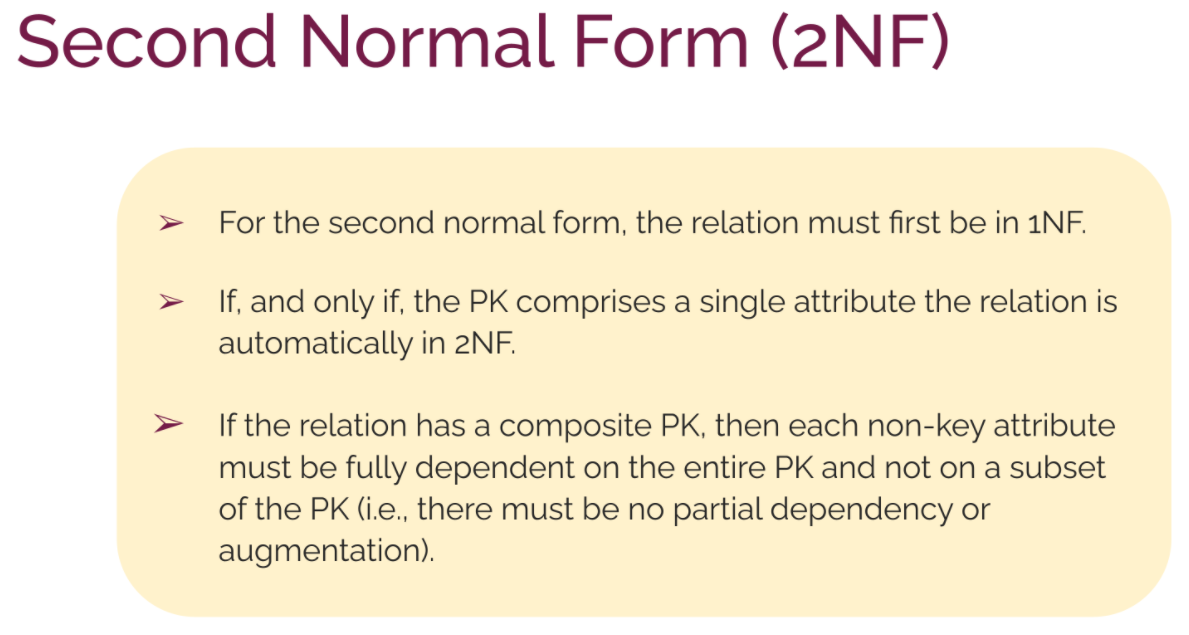
1. Normal formun şartları sağlandığında 2. Normal forma geçebiliyoruz. 2. Normal formun şartları sağlandığında 3. Normal forma geçebiliyoruz.



**1. Normal form.**

* Aynı tablo içinde tekrarlayan kolonlar bulunamaz.
* Her kolonda yalnızca bir değer bulunabilir (bkz. "Semt" kolonu)
* Her satır bir eşsiz anahtarla tanımlanmalıdır (Unique Key - Primary Key)

Bunları sağlamak için: Database deki entitileri (primary key ile) ayırıyoruz. Her bir entitiye uygun bir primary key veriyoruz. Her bir sütunu **atomik yapıda olmasını sağlıyoruz** (yani tek bir bilgiyi ifade edebiliyor olmalı).

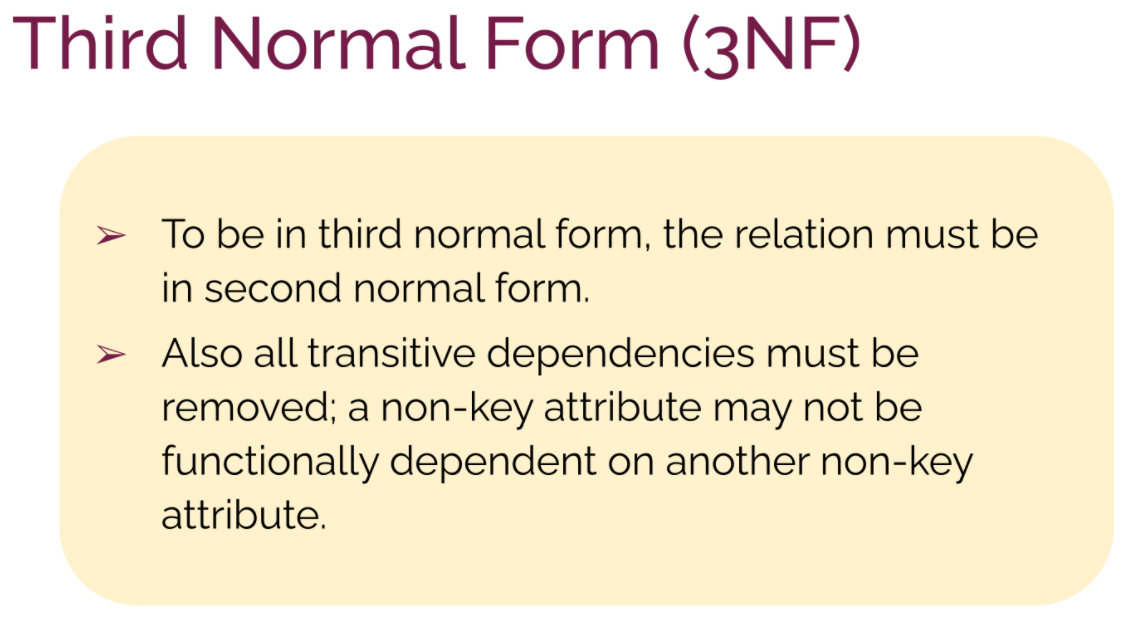


**2. normal form.**

1. Normal form sağlanmış olacak

2. Partial dependency olmayacak.

**Partial dependency ancak komposit key varsa olabileceğinden, tablomuzda komposit key yoksa otomatik olarak ikinci normal form sağlanmış demektir.**



**3. Normal Form:**

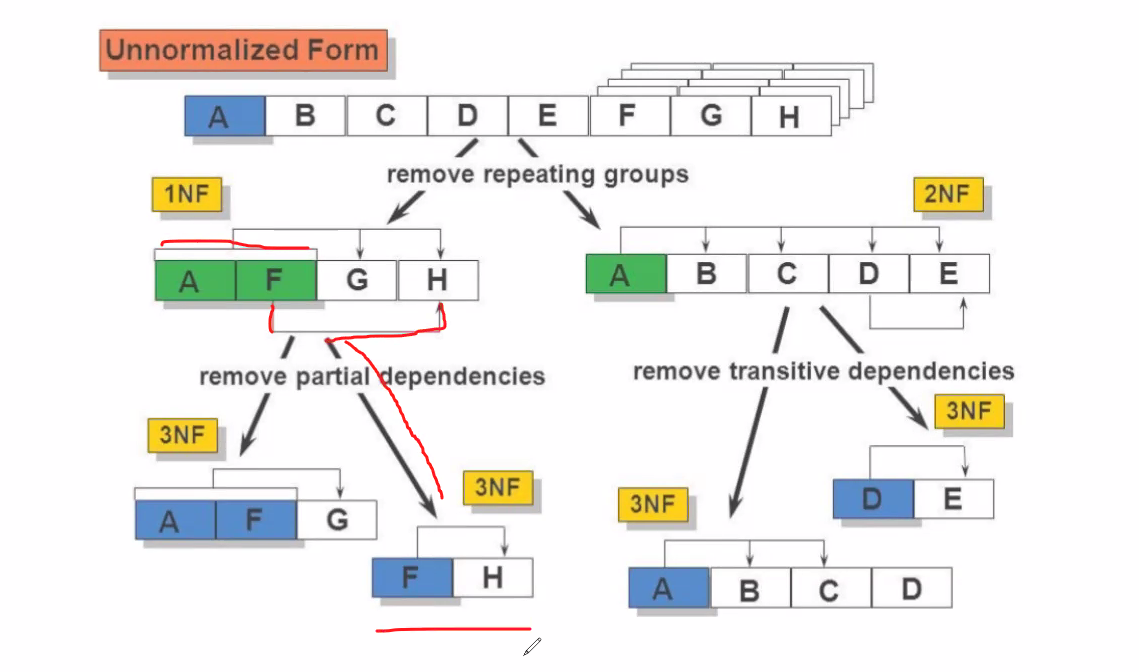
2. Normal form sağlanmış olacak.

Transitive dependency olmayacak.

**( Non-key attibute (sütun), farklı bir non-key sütuna functional dependent olmayacak.)**

**Yani diyor ki; Primary key i bir kenara bırakın, key olmayan diğer sütunlarda bir sütun diğer sütunlara (diğer bir sütuna) bağlımlı (dependent) ise bu bilgiyi bir arada tutamazsın.** Bir sütun primary keye bağımlı olmak zorundadır. Bir sütun bir şeye bir şeye bağlı olacaksa bu primary key olmalı, diğer sütunlardan birine bağlı olamaz.

-------------------------------------------------------------------------

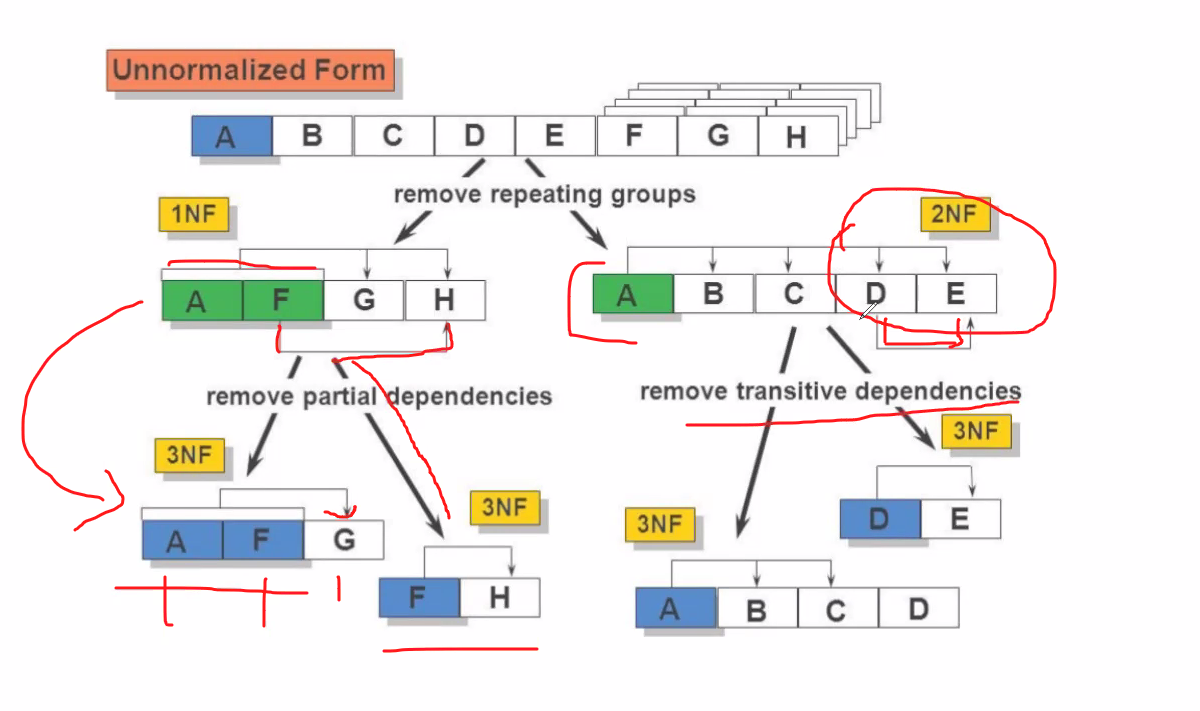


Burada sol taraftaki 1NF tablosunda:

A ve F sütunları birlikte G ile H’ı ( geri kalan sütunları) identify edebiliyor. Aynı zamanda F sütunu H sütununu tek başına identify edebiliyor.

Fakat **bütün sütunlar yalnız bir tek (primary key olan) sütuna fonksiyonel olarak bağlı olabilir.** Burada kompozit key olduğundan tüm sütunlar ancak A ve F’ye birlikte bağlı olabilir**. Başka bir sütun tek başına F’ye bağımlı ise bu bir anomalidir** ve bu durumda diyoruz ki F sütunu sen H ile birlikte ayrıl ve başka bir tabloda (3NF tablosunda) H’yi identify et. Bu tablodan geriye ne kaldı? A, F ve G sütunları kaldı.

Dolayısıyla unnormalize formda olan tabloyu normalize etmiş oluyoruz.



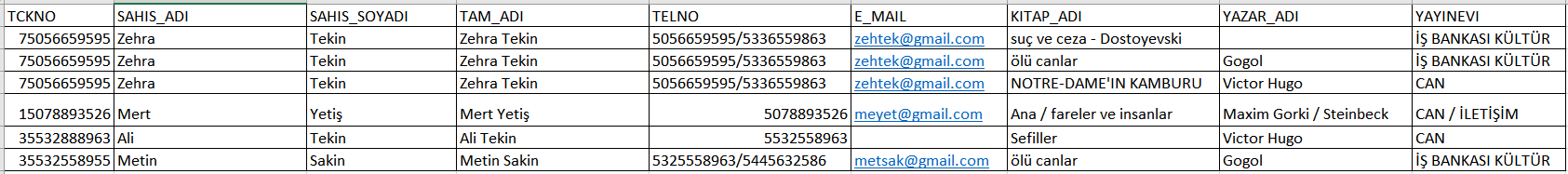
Sağ taraftaki 2NF tablosunda:

Burada A sütunu diğer tüm sütunları identift edebiliyor. Ama primary key olmadığı halde D sütunu da E sütununu identify edebiliyor. Yani E’nin D’ye functional dependent olduğunu (yani fonksiyonel olarak bağlı olduğunu) görüyoruz.

Bu duruma **transitive dependency** diyoruz. Bu da istenmeyen bir durumdur. Çünkü bütün sütunlar yalnızca primary keye funcional dependency olarak bağlı olabilir. Bunlara diyoruz ki siz gidin D’nin primary key olduğu ve E’yi identify ettiği başka bir sütun oluşturun. Bu sütundan geriye ne kaldı? A, B, C, ve D sütunları kaldı. Bu işleme normalizasyon diyoruz.

-------------------------------------------------------------------------

**Bir unnormalize tablo üzerinden bu konuyu pekiştirelim**



İçinde kütüphaneden ödünç kitap alanların bilgileri mevcut olan bu database den relational database oluşturacağız.

Bu tabloda ne gibi problemler var bu açıdan incelediğimizde ;

Bir sütunda şahıs adı bir sütunda soyadı varken başka bir sütunda da şahsın adı ve soyadı beraber yazılmış. Telno sütununda şahsın ikinci telefonu var. İkinci telefonlar slash ile ayrılmış. Kitap adı sütununda suç ve ceza girilmiş ve yanına yazar adı yazılmış ama zaten ayrı bir yazar adı sütunu var. Bir şahsın email bilgisi girilmemiş. Tekrarlar var, boş satırlar var. Bunu first, second, third formlara göre normalize etmem lazım.

Öncelikle bakıyoruz burada bir Id söz konusu değil.

Buradan önce entitileri çıkartmamız gerekecek ve bu entitilere primary key atamamız gerekecek.

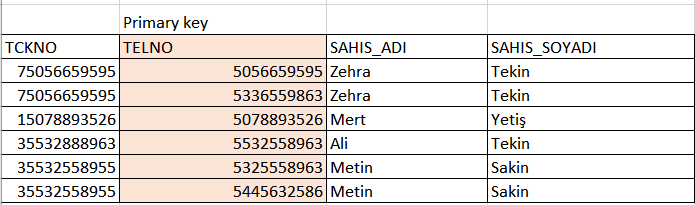
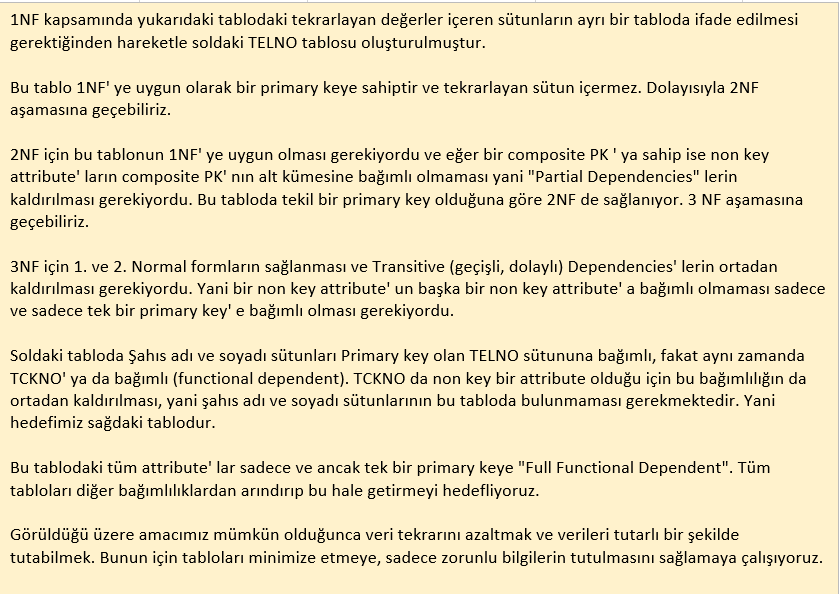
Tespit ettiğimiz entitiler; yayın evi, kitap, yazar, mail adresi, telefon, kişi bunların hepsi birer entiti.

Zehra Tekin isimli kişi çoklamış da çoklamış. Çünkü birden fazla kitap almış ve her kitap için kayıt girilmiş.

Telefon entitisini alıp yeni bir tabloda bunu ifade etmek istiyorum. Telefon bilgisini normalize edilmiş bir entiti tablosunda tutmak istiyorum. Bir telefon numarası tek başına bir şey ifade ediyor mu? Etmiyor. Ne olması lazım? Bu telefon kime ait. Bu bilginin tutulması lazım. O zaman yeni bir tablo oluşturuyorum ve bu tabloda telefon numaraları ve bu numaraların ait olduğu kişiler mevcut olmalı.

Önce entitileri belirliyorum. Telefon entitisi, ad-soyad entitisi, e-mail entitisi gibi.

Telefon entitisini alıp değerlendiriyorum. Telefon bilgisi bana tek bir tablo oluşturabilir mi? Telefon numaraları mevcut ama bunlar tek başına bir şey ifade etmiyor. Bunlar kime ait bunu bilmeliyim. Yeni bir tablo oluşturuyorum. Bu yeni tabloda telefon numaraları ve onların kimeler ait olduğu yer alacak.

Şimdi sağdaki notlarımız üzerinden tablomuzu değerlendirelim.

1.normal form kapsamında, TEL\_NO tablosunu oluştururken unnormalize tablosundaki tekrar eden TAM\_ADI sütununu almadık.

TEL\_NO tablosuna primary key belirlemem lazım. Her bir satırda tekrar etmeyen unique değer olmalı. Tabloya baktığımda TCKNO sütunundaki 1 ve 2. Satırlar telefon numaralarına bağlı olarak tekrarlamış. Zehra Tekin’in ayrı iki tel.no suna karşılık TCKNO su tekrar yazılmış. 1. Satırdaki TC kimlik nosuna baktığımda bit tel no ya işaret ediyor ama 2. Satırdaki aynı TC numarası farklı bir tel no ya işaret ediyor. Eğer bu sütun identify edebiliyor olsaydı ikinci satırda da birinci satırdaki tel numarası olmalıydı. (Yani bir TC kimlik numarasındaki değere bağlı bir değer diğer satırlarda da tekrar etmesi gerekirdi.) Dolayısıyla TCKNO sütunu primary key olamaz.

TELNO sütununa bakalım. Bu entitide her bir telefon numarasına bir satır verildiğini görüyoruz. Ve bu satırlarda telefon numarasının kime ait olduğu bilgisi mevcut. Her satırda farklı bir değer içerdiği için primary key kriterime uyuyor. Bu yüzden TELNO sütununu primary key olarak atıyorum.

2. normal form için bu tablonun 1. Normal forma uyması gerekiyordu. 2. Normal formda partial dependency leri kaldırmam gerekiyordu. Bu tabloda partial dependency sözkonusu değil çünkü tek bir primary key var (komozit key yok) 2. Normal form da sağlandığı için 3.normal forma geçebiliriz.

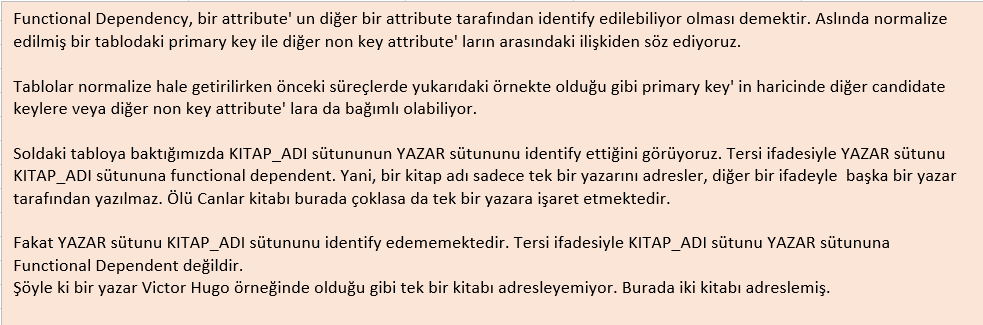
3. normal form için 1 ve 2. Normal formların sağlanması gerekiyor ve trasitive dependencies olmamalı. Yani bir non-key attibute, başka bir non-key attribute’a bağımlı (functional dependent) olmaması gerekiyor. İnceledimizde; her kişinin bir TC kimlik numarası vardır. Dolayısıyla TCKNO sütunundaki her bir kimlik numarası tek bir şahsı adresler. Yani şahıs adı veya soyadı veya ikisi birden TCKNO sütununa functional dependent’tır! Bu trasitive dependency demek. Bu yüzden bu iki sütunu yeni bir tabloya koyacağız ve kişi bilgilerini içeren yeni bir entiti tablosu oluşacak, primary key’i ise TCNO sütunu olacak. Tablomuzdan geriye kala kala Primary key (TELNO) ve TCKNO kaldı.



Bu tablodaki telefon numarasının isim olarak kime ait olduğunu bulmak istiyorsam ne yapacağım? TCKNO bilgisi üzerinden şahıs tablosuna ulaşacağım (join yapacağım) ve ordan ismi çekeceğim.

**Şimdi partial dependency konusunu bir örnekle inceleyelim:**

****

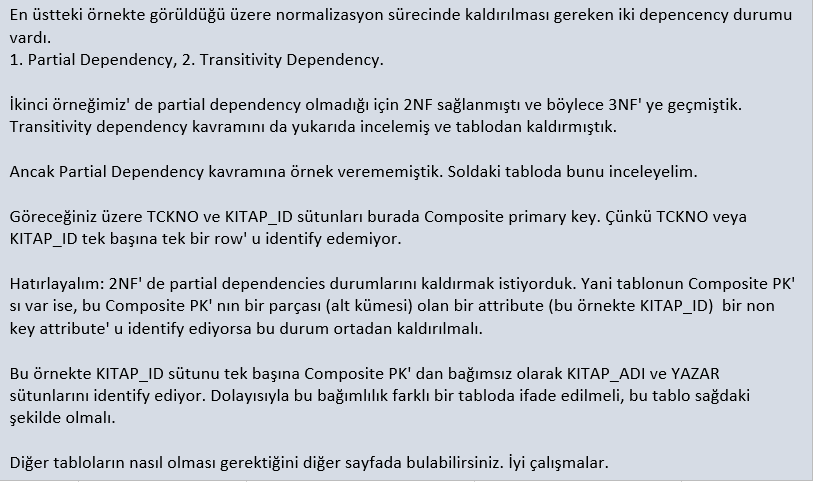
****

Bir konuyu değerlendirirken kendi içinde değerlendiriyoruz. Mantığımıza uydurmaya çalışmıyoruz. Elimizdeki verilere göre değerlendiriyoruz. Olması gereken gerçek dünyadaki tüm örnekleri temsil edemeyeceğimiz için burda belli bir kısmını aldık ve örnek kümemiz üzerinden konuyu detaylandırmaya çalışıyorum. Dolayısıyla şuanki konuyu gerçek hayattan bağımsız olarak değerlendiriyoruz. Yalnızca bize verilen yukardaki tablodaki (unnormalize olan tablodaki) verileri değerlendiriyoruz.

KITAP\_ADI sütununun YAZAR sütununu identify ediyor. Öyle mi peki? Kitap adı sütunu Yazar sütununu idetify ediyorsa her bir kitap adı tek bir yazarı adresliyor olmalıydı. Suç ve Ceza’ya bakıyorum. Kitap adı birden farklı bir yazara ait olamaz diye düşünüyorum. Ana’yı Maxim Gorki yazmış. Bu değişir mi? değişmez. Aşağıya iniyorum, Ölü Canlar kitabı var. Bunlar tekrarlamış fakat tek bir yazarı adresliyor. 2. Satırda da karşısında Gogol var 8. Satırda da Gogol var. Yani Ölü Canlar Gogol’u identify ediyor.

Peki tam tersinden bakarsak YAZAR sütunu KITAP\_ADI sütununu identify edebiliyor mu? Bir yazarın birden fazla kitabı olabilir. Mesela 5 ve 6. Satırlardaki Victor Hugo bir kitabı identify edemiyor, bu yazarın karşısında 2 ayrı kitap adı var. Dolayısıyla KITAP\_ADI sütunu YAZAR sütununa functional dependent değil.

Bu tablomuzda partial dependency olmadığı için burada partial dependency’ye örnek veremedik. Şimdi başka bir tablodan partial dependency’ye bakalım:

Bu tabloya baktığımızda bize şunu anlatıyor. Kitap\_id ler var ve bu kitap\_id lere ait kitapları ödünç alan kişilerin TCKNO ları var. Yani bu ödünç alma tablosunda hangi kişi hangi kitabı ödünç almış bilgisi mevcut.

TCKNO ile KITAP\_ID eşleşmesi beraberce diğer tüm sütunları identify etmesi gerekiyor. Fakat bakalım TCKNO bu tabloyu identify edebiliyor mu? Sonu 9595 olan TCKNO karşısında farklı KITAP\_ID leri var. Bir TCKNO yalnızca bir KITAP\_ID yi adresleyemiyor, tekrar ediyor. Dolayısıyla primary key özelliklerini karşılayamıyor. Demek ki TCKNO sütunu da tek başına primary key olamıyor

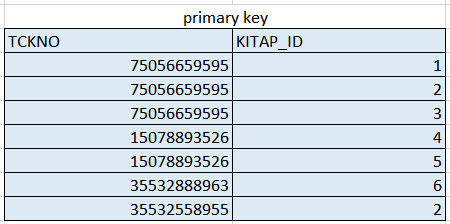
Şimdi tam tersi KITAP\_ID sütunu TCKNO sütununu identify edebiliyor mu buna bakalım. Bu sütunda 2 nolu KITAP\_ID, farklı iki TCKNO yu adreslemiş. Demek ki farklı tarihlerde bu kitabı farklı iki kişi almış. Bu duruma 2 nolu ID, tek bir TCKNO yu adresleyememiş oluyor. Yani KITAP\_ID sütunu TCKNO sütununu identify edemiyor. KITAP\_ID sütunu da tek başına primary key olamıyor.

Bu durumda TKCKO ile KITAP\_ID sütunları beraber kompozit olarak primary key olmak zorunda.

Peki bu kompozit key’in sütunlarından biri non\_key olan bir sütunu identify ediyor mu? (partial dependency var mı?)

Baktığımda her bir kitap adına farklı bir kitap id verilmiş. Örnek olarak 2.satırdaki 2 olan kitap\_id nosu Ölü Canlar kitabını ve Gogol yazarını adresliyor, son satırdaki 2 olan kitap\_id nosu da yine Ölü Canlar kitabını ve Gogol yazarını adresliyor. Demek ki 2 no lu kitap id’si ölü canlar kitabı ile Gogol yazarını identify ediyor. Yani KITAP\_ADI ile YAZAR sütunları KITAP\_ID sütununa dependent. Bu demektir ki iki non-key sütun, composit key’in bir alt kümesine (bir sütununa) functional dependent, yani bu tabloda partial dependency oluşmuş oluyor.

O zaman bu sütunlardan yeni bir tablo oluşturuyorum ve önceki tablodan elimde bu aşağıdaki sütunlar kalıyor. Yani tablonun olması gereken hali aşağıda olduğu gibidir:

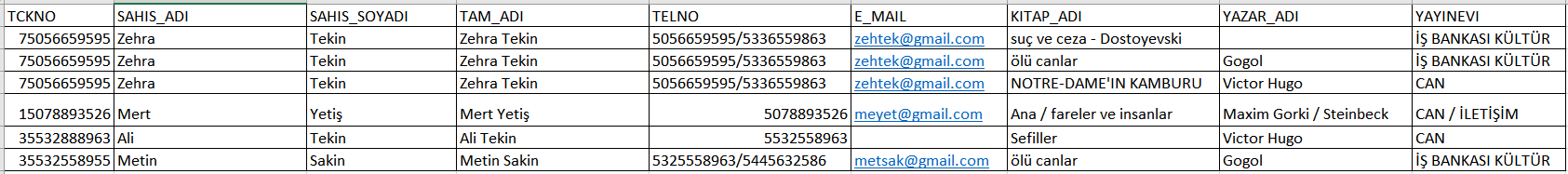


-------------------------------------------------------------------------------------------------

Hatırlarsanız az önce Phone entitisi oluşturduk ve aynı tablo içerisinden bir de Person entitisi çıktı. Bu iki entiti arasında nasıl bir ilişki var, bunu nasıl ifade edebiliriz ona bakacağız.

Phone entitisi içinde bir phone\_number vardı bir de FOREIGN key olarak SSN (sosyal security number) vardı. Nerden geliyordu bu SSN? Person tablosundan geliyordu. Bu durumda Phone entitisi içerisinde SSN var ve bunun üzerinden bu tablo Person tablosu ile bir ilişki kurabiliyor. Şimdi person tablosundan phone tablosuna gidiyor olucam.

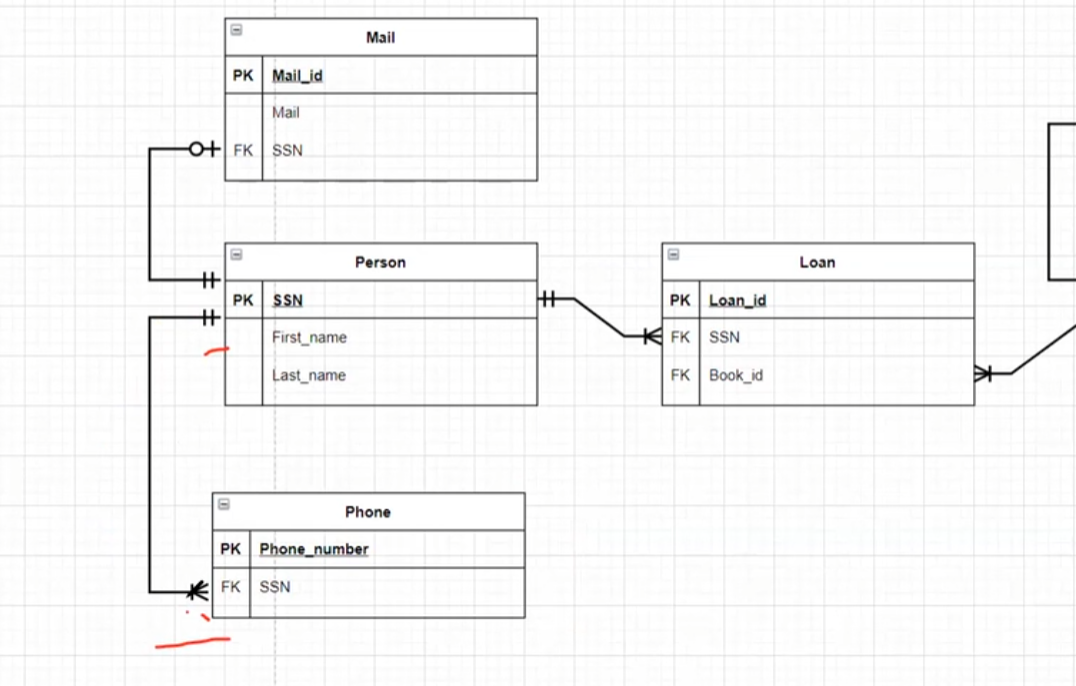
Tablomuzu hatırlayalım. Bir kişinin birden fazla telefon numarasına sahip olma durumu vardı. Mesela Zehra Tekin’in iki telefon numarası vardı.



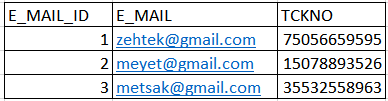
Aşağıdaki diyagramda Phone ve Person tablolarındaki relation işaretine baktığımızda bir kişinin bir veya birden fazla telefon numarası olabilir. Yani one to many durumu var (Bir kişinin telefon numarası olmaya da bilir ama biz gerçek dünyaya bakmıyoruz, elimizdeki dataya bakıyoruz. Demek ki datasetimizde her kişinin telefon numarası var)

Tersten bakarsak; bir telefon bir kişiye ait olabilir. Yani many to one. Bir telefonun kimseye ait olmama durumu söz konusu mu? Değil.

Dolayısıyla burası  yuvarlak ( o ) değil.



Aşağıda mail tablomuz var. Bakın normalize edilmiş tabloda her bir maile bir mail id verilmiş ve bunun kime ait olduğu belli. Bakıyorum 3 tane kişinin mail adresi varmış. Benim datamda toplam kaç kişi var? 4 kişi var. Ama sadece 3’ünün mail adresi var. Demek ki her kişinin mail adresi olmayabilir. Ve baktığımda her kişinin 1 tane mail adresi var.



Şimdi diyagrama tekrar geliyoruz:

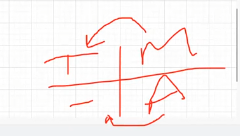
Bir kişinin hiç mail adresi olmayabilir veya en fazla 1 mail adresi olabilir. (gerçek dünyayı hesaba katmıyoruz) Burada kaz ayağı işareti olsaydı bir kişinin birden fazla mail adresi de olabilir olurdu.



Mail tablosundan Person tablosuna doğru bakarsak; bir mail, bir kişiye ait olabilir veya en fazla bir kişiye ait olabilir. İsimsiz mail var mı? İsimsiz mail yok. Person sütununda TCKNO su boş olan herhangi bir mail var mı? Yok. Her mailin ait olduğu bir kişi var benim verimde.

Eğer aşağıdaki gibi bir durum olsaydı. Yani “A” mail’inin karşısında TC no boş olsaydı. Bu durumda bir mail bir kişiye ait olabilirin yanında kimseye ait olmayabilir durumu da olurdu. (kimseye ait olmayan ama sistemimde mevcut bir rezerv mail var diyecektim.)

Bu durumda Mail tablosundan Person tablosuna giden ilişkide ( o ) opsiyon durumu olacaktı.



--------------------------- THE END ------------------------------------------------