**FUNCTIONAL DEPENDENCY, PARTIAL DEPENDENCY, TRANSITIVE DEPENDENCY, NORMALIZATION, CHAR/ VARCHAR, CONSTRAINTS, DROP, TRUNCATE TABLE, DELETE**

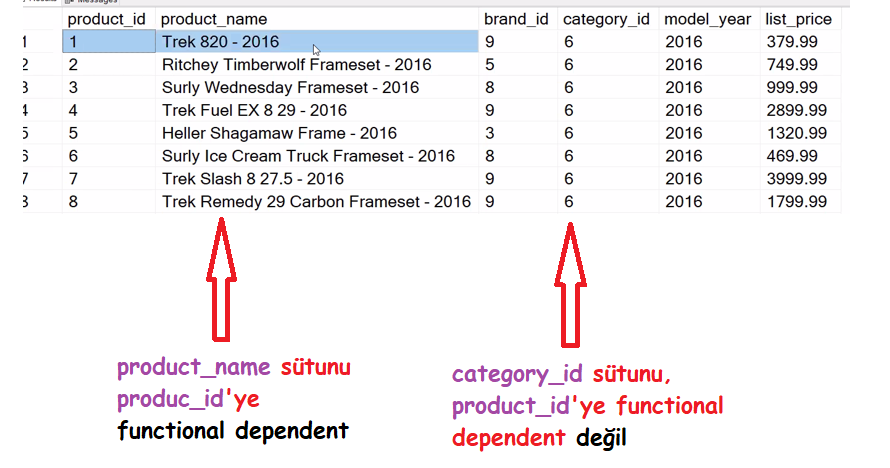
**FUNCTIONAL DEPENDENCY :**

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

**Bir sütunun başka bir sütunu "*identify"* etmesi demek, her bir değerine ait diğer sütunda tek bir değere işaret etmesi demek.**

Aşağıdaki örnekte; Product\_name sütunu product\_id sütunu tarafından tanımlanabildiği (**identify edildiği**) için product\_name sütunu product\_id sütununa **functional olarak bağlı**diyoruz. Buna **FUNCTIONAL DEPENDENCY**diyoruz.

Yani; *product\_name, product\_id’ye functional dependent çünkü her bir product name'in yalnızca bir tek product id'si var. Fakat category\_id, product\_id ye functional dependent değil çünkü birden çok “6” değeri farklı product\_id’lerde tanımlı!!*

****

**Kural : Bir tablolda tüm sütunlar Primary Key'e FULL FUNCTIONAL DEPENDENT olmalı**.

**Kural : Primary Key dışındaki sütunlar başka bir sütuna functional dependent OLMAMALI!**

**PARTIAL DEPENDENCY:**

**Composit Key'**in 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.**

**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.

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

**1. Normal form.**

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**.

**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 dışındaki (key olmayan) diğer sütunlarda bir sütun 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.

**CHAR ile VARCHAR ARASINDAKİ FARK:**

CHAR ile karakter uzunluğunu 200 olarak belirlediğinizde sonra siz bir satıra ister 5 karakter girin ister 200 girin CHAR her zaman hafızada 200 karakter yer tutar.

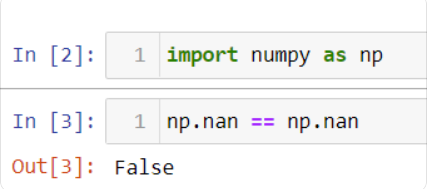
Ama VARCHAR'a diyelim 200 girdiniz. Ama bir satıra 10 karakter yazdınız VARCHAR hafızada girdiğiniz karakter kadar yer kaplar. 200 karakter yer kaplamaz!

**CONSTRAINTS:**

**Null --->** (Null içerebilir veya içermeyebilir)  
**Not Null --->** (Boş bırakılamaz)  
**Unique --->**  (Bir değer ancak bir kere girilebilir)  
**Default --->** (Hücre boş kaldığı durumda Default olarak önceden belirlenen değeri alıyor)  
**Identity --->** (sütunun her hücresi için id değer atar. Girilen her kayıt için bir id nosu atar)  
**Primary Key** ---> (Bir değerden iki tane olamaz (Unique olmalı). NULL’da alamaz)  
**Foreign Key --->** (Bir değerden iki tane olamaz, null da alamaz. Bir Foreign Key başka bir tablonun Primary Key'idir)  
**Check --->** (Sütuna alınacak değerlere bir filtre uyguluyoruz. Mesela yaş, eğer yaş kısıtı olan bir field oluşturacaksak bu constrainti kullanacağız)

NULL değerlerin bir değeri olmadığı için onları =, <, <> gibi Comparision operatörleri ile test edemezsiniz. Null olup olmadığını test için **IS NULL**ve **IS NOT NULL**operatörlerini kullanıyoruz.

Tıpkı ne gibi? NaN değerlerin belirli bir değeri olmadığı ve iki NaN'ın birbirine eşit olmadığı gibi



**SQL komutları 4 ana gruba ayrılır:**

**Veri Sorgulama Dili (Data Query Language - DQL)**

(SELECT komutu ile başlayan query'ler..)

**Veri Kullanma Dili (Data Manipulation Language - DML)**

(Veritabanlarında bulunan verilere işlem yapılır. INSERT, UPDATE, DELETE)

**Veri Tanımlama Dili (Data Definition Language - DDL)**

(veritabanı ve tabloları oluşturma, değiştirme ve silme işlemleri yapılır. CREATE, ALTER, DROP)

**Veri Kontrol Dili (Data Control Language - DCL)**

(kullanıcılara veritabanı ve tablolar için yetki verilir veya geri alınır. GRANT, REVOKE)

**DROP :**

* DROP yapısı ile indexler, alanlar, tablolar ve veritabanları kolaylıkla silinebilir.
* DELETE yapısı ile karıştırılabilir. DELETE ile sadece tablomuzdaki KAYITLARI silebiliriz.
* Eğer TABLO veya VERİTABANINI silmek istiyorsak DROP yapısını kullanmamız gerekmektedir.

**TRUNCATE TABLE:**

Eğer tablomuzu değil de sadece içindeki kayıtları silmek istiyorsak yani tablomuzun içini boşaltmak istiyorsak..

**DELETE:**

* Burada dikkat edilecek nokta WHERE ifadesi ile belli bir kayıt seçilip silinir.
* Eğer WHERE ifadesini kullanmadan yaparsak tablodaki bütün kayıtları silmiş oluruz.
* **TRUNCATE**yapısında parametre girilmez direkt olarak tüm kayıtları siler.  Yeni kayıt yapılırsa numarası 1 den başlar.
* **DELETE** ile bütün kayıtları sildiğimiz zaman otomatik numara sırası baştan başlamaz. Örneğin 150 kayıt silindiğinde ve yeni kayıt eklediğimizde bu 151 olur.

**SQL'de köşeli parantez kullanımı:**

**Köşeli parantez içindeki herhangi bir şey tek bir tanımlayıcı olarak kabul edilir.**örneğin sütun adı içinde boşluk varsa ve bunu köşeli parantez içine almazsanız bunları iki ayrı keyword olarak algılar ve hata verir. Owen hocanın yöntemi garantidir

SQL, tablo adlarının, sütun adlarının veya diğer meta veri nesnelerinin aşağıdakileri içermesine izin vermek için tanımlayıcıların sınırlayıcılarını destekler:

* SQL ayrılmış kelimeler: "Sipariş"
* Boşluk içeren kelimeler: "Order qty"
* Noktalama işaretlerini içeren kelimeler: "Order-qty"
* Uluslararası Karakter içeren kelimeler
* Büyük/küçük harf duyarlı sütun adları: "Sipariş" ile "sipariş"