**SQL SESSION – 1**

**PRIMARY KEY:**  
\* Satırlara ait değerlerin karışmaması adına bu alana ait bilginin tekrarlanmaması gerekir. (**unique olmalı)**  
\* Temel işlevi, verilerin hangi satıra dizileceğine (veya hangi satırda verilerin düzenleneceğine) karar vermesidir.  
\* Genelde sayılar Primary Key olarak seçilirler, ancak zorunluluk değildir.  
\* Primary key değeri**boş geçilemez** ve NULL değer alamaz.  
\* İlişkisel veri tabanlarında (relational database management system) **mutlaka Primary Key olmalıdır**.

**FOREIGN KEY:**  
\* Bir veri tablosuna girilebilecek değerleri başka bir veri tablosundaki alanlarla **ilişkilendirmeye**yarar.  
\* **Özetle, başka bir tablonun Primary Key'inin bir diğer tablo içerisinde yer almasıdır.** (Yer aldığı tabloda buna Foreign Key diyoruz)  
\* Çoğunlukla bir ana tablo (parent) ile alt tablonun (child) ilişkilendirilmesinde kullanılır.  
\* Bir tabloda **"id"** alanına PRIMARY KEY uygulayabiliriz. **Ancak aynı tablodaki başka bir alan, farklı bir tablodaki kayda bağlı çalışabilir**  
\* **İşte bu iki tablo arasında bir bağ kurmak gerektiği durumlarda FOREIGN KEY devreye giriyor.**  
\* Böylece tablolar arası veri akışı daha hızlı olduğu gibi ileride artan kayıt sayısı sonucu veri bozulmalarının önüne geçilmiş olunur.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Bir şehir tablosu yaparsınız ve şehirleri tabloda id numaralarıyla tutarsınız, ve üyeler tablosunda hangi üyenin hangi şehirde bulunduğunu id bağlantısıyla yaparsınız ki üye sayısı arttıkça üyeler tablosundaki şehir kolonunda string değer tutmadan sadece int değer tutarak verinin çok yer kaplamamasını sağlarsınız.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ER modeli;** oluşturulacak veritabanı nesneleri arasında **ilişki** kurarak nesnelerin **özelliklerini**ortaya koyar.

**Entity relationship diagram (ERD)**: Bir sistemdeki objeler arasındaki ilişkileri gösterir.

**Entity:** veritabanında oluşturulacak **nesneleri** temsil eden yapılardır. Entitilere database'deki **tablolar**örnek verilebilir. ER Diyagramının temelini oluştururlar.  
Database'deki entitileri primary key ile ayırıyoruz. Her bir entitiye uygun bir primary key veriyoruz.

**Nitelik (Attiribute):**  
Attiribute, ER entitilerin  sahip olduğu parçaları oluşturan bileşenlere denir. (Örnek olarak tablo sütünları (alanları) ).

**İlişki (Relationship):**  
Relationship, **entitiler arasında kurulan fiziksel ve mantıksal bağlantıları** temsil eden yapılardır.  ER diyagramlarında entitilerin arasındaki ilişkileri belirler.

**ER Diyagram oluşturma process'inde sırayla;**

* İlk olarak entitiler belirlenir.
* Daha sonra her **entitinin nitelikleri (attribute) belirlenir.**
* Belirlenen nitelikler arasında oluşturulan primary key nitelik altı çizili şekilde belirtilir.
* Bu işlemler yapıldıktan sonra entitiler arasında **ilişkiler (relation) oluşturulur.**
* Daha sonra çizilen bu ER diagramın **normalizasyon**kurallarına uygunluğu incelenir.
* Uygun olmayan kısımlar düzeltilerek ER diagramı tamamlanır.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PRIMARY KEY:**

* **Bir tabloyu tek başına identify etme (tanımlama) yeteneğine sahip keydir.**
* Boş geçilemez, Null değer içeremez.
* **Satırlara ait değerlerin karışmaması adına bu alana ait bilginin tekrarlanmaması gerekir.**
* Genelde sayılar Primary Key olarak seçilirler.

**FOREIGN KEY:**

* Bir veri tablosuna girilebilecek değerleri başka bir veri tablosundaki alanlarla **ilişkilendirmeye**yarar.
* **Özetle, başka bir tablonun Primary Key'inin bir diğer tablo içerisinde yer almasıdır.** (Yer aldığı tabloda buna Foreign Key diyoruz)

**COMPOSITE KEY:**  
Bir tabloda Primary Key olabilecek ve tek başına bu tabloyu identify edebilecek bir alan yok ise; **ancak ikisi alan beraberce bu tabloyu identify edebiliyorsa**bu iki alana beraber **Composit Key** diyoruz.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Entiti Relationship Diagram (ERD)**’da database’in tablolarının **normalize edilmiş halini** ve **bunlar arasındaki ilişkileri gösteriyoruz.**

ERD bize normalize edilmiş bir database’in yapısını yansıtıyor.

Eğer ralational database oluşturacaksak bu **normalizasyon mutlaka yapılmalıdır**

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 son derece dikkat etmek icap ediyor.

Bunun önüne geçmek için **relational database**kavramı oluşmuş ve tüm sütunların bir tabloda toplanması **ANOMALİ**olarak görülmüş ve bundan vazgeçilmiş.

**Anomalilerin giderilmesi işlemine NORMALİZASYON denilmektedir**.

Özetle normalizasyonun **iki temel amacı var**:

1. Veri t~~a~~banında veri tekrarlarını ortadan kaldırmak
2. Veri tutarlılığını (doğruluğunu) artırmak.