**INNER JOIN – JOIN Kullanımı**

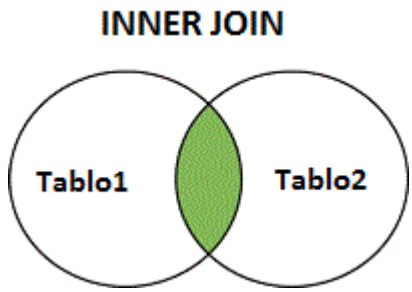
İki adet tablomuzdaki kayıtları belli bir kritere göre birleştirmek için INNER JOIN komutu kullanılır.

**INNER JOIN Kullanım Biçimi**

```
SELECT alan_ad(lari)
FROM tablo1 INNER JOIN tablo2
ON tablo1.alan_adi=tablo2.alan_adi
```

veya

```
SELECT alan_ad(lari)
FROM tablo1 JOIN tablo2
ON tablo1.alan_adi=tablo2.alan_adi
```




MD.JOINSAMPLES - dbo.musteriler | SQLQuery1.sql - not

| | id | Ad_Soyad | sehir |
|---|---|---|---|
| ▶ | 1 | Ali Veli | Van |
| | 2 | Hasan Yılmaz | Mersin |
| | 3 | Gül Çekmez | Çanakkale |
| | 4 | Hakan Yıldız | İstanbul |
| * | NULL | NULL | NULL |

MD.JOINSAMPLES - dbo.satislar | MD.JOINSAMPLES - db

| | id | satilan_Mal | satis_fiyat |
|---|---|---|---|
| ▶ | 1 | Buzdolabı | 1500,0000 |
| | 1 | Bilgisayar | 3000,0000 |
| | 2 | LCD TV | 3200,0000 |
| | 1 | Çamaşır Mak. | 1200,0000 |
| | 5 | ürün | 2000,0000 |
| * | NULL | NULL | NULL |

```
SELECT *
FROM musteriler JOIN satislar
ON musteriler.id = satislar.id

SELECT Ad_Soyad,satilan_Mal
```

```
FROM musteriler JOIN satislar
ON musteriler.id = satislar.id
```

```
SELECT Ad_Soyad,satilan_Mal,musteriler.id AS id_no
FROM musteriler JOIN satislar
ON musteriler.id = satislar.id
ORDER BY id_no ASC
```

**LEFT JOIN – LEFT OUTER JOIN Kullanımı**

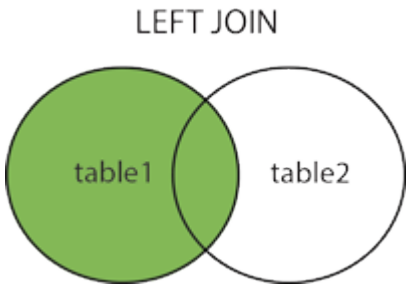
LEFT JOIN ile iki adet tablomuzdaki kayıtları belli bir kritere göre birleştirebilir. Burada asıl olan birinci tablondaki kayıtlardır. İkinci tablodan sadece birinci tabloda olan kayıtlar alınır. İkinci tabloda olupta birinci tabloda olmayan alanların değeri boş (NULL) olarak gelecektir.

**LEFT JOIN Kullanım Biçimi**

```
SELECT alan_ad(lari)
FROM tablo1 LEFT JOIN tablo2
ON tablo1.alan_adi=tablo2.alan_adi
```

veya

```
SELECT alan_ad(lari)
FROM tablo1 LEFT OUTER JOIN tablo2
ON tablo1.alan_adi=tablo2.alan_adi
```




MD.JOINSAMPLES - dbo.Ogretmen

| | OgretmenID | OgretmenAd_... |
|---|---|---|
| ▶ | 1 | HASAN KALIN |
| | 2 | ALİ GÜR |
| | 3 | CEVDET DURMUŞ |
| | 4 | HAKKI VAROL |
| * | NULL | NULL |

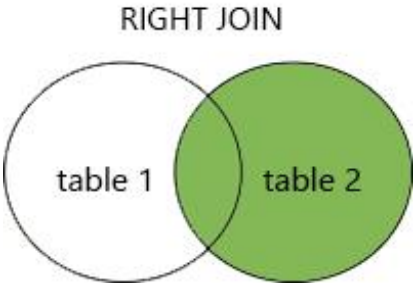
MD.JOINSAMPLES - dbo.Ders — MD.JOINSAMPLES - dbo.Ogretmen

| | DersID | DersAdi | HaftalikSaat | OgretmenID |
|---|---|---|---|---|
| ▶ | 1 | TÜRKÇE | 6 | 1 |
| | 2 | FEN | 5 | 2 |
| | 3 | MATEMATİK | 5 | 5 |
| | 4 | COĞRAFYA | 2 | 8 |
| * | NULL | NULL | NULL | NULL |

```
SELECT *
FROM Ogretmen o LEFT JOIN Ders d
ON o.OgretmenID = d.DersID
```

**RIGHT JOIN –OUTER JOIN Kullanımı**

RIGHT JOIN ile iki adet tablomuzdaki kayıtları belli bir kritere göre birleştirebilir. Burada asıl olan İkinci tablondaki kayıtlardır. Birinci tablodan sadece ikinci tabloda olan kayıtlar alınır. Birinci tabloda olupta ikinci tabloda olmayan alanların değeri boş (NULL) olarak gelecektir.




MD.JOINSAMPLES - dbo.Personel | MD.JOINSAMPLES - dbo.Ders

| | id | Adi_Soyadi | Dep_id | sehir |
|---|---|---|---|---|
| ▶ | 1 | Hakan Adalı | 2 | İstanbul |
| | 2 | Cezmi Pekcan | *NULL* | Ankara |
| | 3 | Canan Balcı | 4 | Burdur |
| | 4 | Metin Türker | 1 | Çanakkale |
| | *NULL* | MUJDAT | 5 | Ankara |
| * | *NULL* | *NULL* | *NULL* | *NULL* |

MD.JOINSAMPLES - dbo.Departman

| | id | Departman_Ad |
|---|---|---|
| ▶ | 1 | Muhasebe |
| | 2 | Bilgi İşlem |
| | 3 | Satış |
| | 4 | Pazarlama |

```
SELECT *
FROM Personel P  RIGHT JOIN Departman D
ON P.Dep_id = D.id
```

**FULL JOIN – FULL OUTER JOIN Kullanımı**

İki tablodaki tüm veriyi kesişimde dahil olarak verir.

```
SELECT *
FROM Personel P  FULL JOIN Departman D
ON P.Dep_id = D.id
```

## UNION Kullanımı

UNION ile iki adet tablomuzdaki seçeceğimiz alanları birleştirerek tek bir tablo alanıymış gibi kullanabiliriz. Union ile iki tablodaki alanlar birleştirilirken tekrarlayan kayıtlar bir defa alınır. Eğer tekrarlayan kayıtların alınması isteniyorsa UNION ALL kullanılmalıdır.

**UNION Kullanım Biçimi**

```
SELECT alan_ad(lari) FROM tablo1
UNION
SELECT alan_ad(lari) FROM tablo2
```

**UNION ALL Kullanım Biçimi**

```
SELECT alan_ad(lari) FROM tablo1
UNION ALL
SELECT alan_ad(lari) FROM tablo2
```

Görüleceği üzere iki tane SELECT ifadesi kullanılmaktadır. Yani iki ayrı sorgu yapısını UNION ile birleştirmiş oluyoruz. Burada dikkat edilecek olan nokta Select ifadesinden sonra yazılacak alan sayısı her iki sorgu ifadesinde de aynı olmalıdır. Alan adları farklı olabilir. Yani birinci select ifadesinde Şehir alanı kullanılırken diğer select ifadesinde Adres alanı kullanılabilir. Sonuçta anlamsız bir veri çıkabilir ancak yapı bu şekilde çalışmaktadır. Alanları birleştirirken yazım sırasına göre birleştirme yapmaktadır. Yani birinci Select ifadesinden sonra Adi_soyadi, Sehir yazıldıysa, çekilen verinin anlamlı olması için ikinci select ifadesinden sonra da Adi_soyadi, Sehir şeklinde yazılması gerekmektedir. Eğer ikinci bölüme Sehir, Adi_soyadi yazılırsa birinci tablodan adi_soyadi alanındaki veriler ile ikinci tablodan Sehir alanındaki veriler birleştirilir. Ancak bazı SQL editör programları böylesi bir durumun önün geçmek için kendi içlerinde kontrol mekanizması kurarak kullanıcıyı uyarabilmektedirler.

***Örnek Tablo Uygulaması:***
Örnek olarak aşağıdaki gibi Personel isimli tablomuz olsun.

| id | Adi_soyadi | Sehir |
|---|---|---|
| 1 | Salih ESKİOĞLU | İstanbul |
| 2 | Ayhan ÇETİNKAYA | Kocaeli |
| 3 | Serkan ÖZGÜREL | Erzincan |
| 4 | İlhan ÖZLÜ | İstanbul |

İkinci tablomuz olan Musteriler ise aşağıdaki gibi olsun.

| id | Adi_soyadi | Sehir |
|---|---|---|
| 1 | Veysi Yamlı | Van |
| 2 | Sırrı Derman | Mersin |

***Örnek1:***

```
SELECT Sehir FROM Personel
UNION
SELECT Sehir FROM Musteriler
```

Bu kod ile iki tabloda Sehir alanlarındaki veriler tekrar edenler bir defa alınmak suretiyle birleştirilmiş olunur. Dikkat edileceği üzere PErsonel tablosunda iki tane İstanbul bulunmaktadır.

Çıktısı:

| **Sehir** |
|---|
| İstanbul |
| Kocaeli |
| Erzincan |
| Van |
| Mersin |

**NORMALİZASYON**

Normalizasyon yani diğer adı ile Ayrıştırma, veritabanlarında çok fazla sütun ve satırdan oluşan bir tabloyu tekrarlardan arındırmak için daha az satır ve sütun içeren alt kümelerine ayrıştırma işlemidir. Bunların tabi ki kuralları vardır bu kurallara uyduğumuzda her tabloda aynı sütun ve satırları tekrar etmemiş olacağız ve veritabanındaki verilerimiz sağlıklı kullanıma uygun olacaktır. Şimdi hazırsanız başlayalım.

**Amacı:**
Gereksiz veri tekrarını ortadan kaldırarak, veri fazlalığını en aza indirmektir. Veri tekrarı, veri sapmasına yol açar.Bu da veri bütünlüğünün bozulmasına neden olur.

- Veri bütünlüğünün sağlanması
- Uygulamadan bağımsızlık
- Performansı artırmak

**Avantajları:**
- Veri bütünlüğünü sağlar.
- Verimli bir veri yapısı sunar.
- Gereksiz veri tekrarını engeller ve min. Alan kullanılır ve yerden tasarruf sağlar
- Saklanan veri daha anlaşır hale gelir.
- Hızlı sorgulama imkanı verir.

**Normalizasyon Kuralları**
Normalizasyonun her bir kuralı yani seviyeleri normal form olarak adlandırılır. Bu seviyeler gereksiz veri tekrarlarını ne derecede engellediği ve tutarlılığı ne kadar sağladığına bağlı olarak derecelendirilir. Seviye yükseldikçe veri tutarlılığı artar, veri tekrarı düşer.

**Örnek : Aşağıda excel tablosu olarak verilmiş olan ham veri tablosunu normalize ederek veritabanı olarak oluşturalım.**

| Adı Soyadı | Doğum Yılı | Doğum Yeri | Baba Adı | Mail Adresleri |
|---|---|---|---|---|
| Ali veli | 1.01.1976 | İstanbul | Ali | ali@ali.com, alim@veli.com |
| Ali Deli | 2.01.1976 | İstanbul | Hasan | hasan@ali.com, kasan@veli.com |
| Ali veli | 3.01.1976 | İstanbul | Ali | ali@ali.com, alim@veli.com |
| Ali veli | 4.01.1976 | İstanbul | Ali | ali@ali.com, alim@veli.com |

Yukarıdaki excel tablosunda 1NF ile normalizasyon kurallarını uygularsak Doğum yeri alanında veri tekrarlarını engellemek amacıyla iller tablosu oluşturulabilir. Bunun dışında mail adresleri aynı kolon içerisinde tanımlanmış ya da mail adresi1, mail adresi2… şeklindede tanımlanabilirdi. Bu durumu düzeltmek için mailler tablosu da oluşturulabilir.

Aşağıda normalizasyon uygulanmış halde veritabanı içeriği olarak görünebilir.

MD.NORMALIZASYON - dbo.Kisiler | MD.JOINSAMPLES - dbo.Departman | MD.JOINSAMPLES - dbo.Personel

| | KisilD | Adi | Soyadi | DogumYili | DogumYeri | BabaAdi |
|---|---|---|---|---|---|---|
| ▶ | 1 | Müjdat | Durmaz | 1978-10-04 | 34 | Kasım |
| | 2 | Hasan | Başkıran | 1976-05-03 | 35 | Ali |
| * | NULL | NULL | NULL | NULL | NULL | NULL |

MD.NORMALIZASYON - dbo.Iller | MD

| | illD | ilAdi |
|---|---|---|
| ▶ | 1 | Adana |
| | 2 | Adıyaman |
| | 3 | Afyonkarahisar |
| | 4 | Ağrı |
| | 5 | Amasya |

MD.NORMALIZASYON - dbo.Mailler | MD.NORMALIZ

| | mail_ID | mailAdresi | KisilD |
|---|---|---|---|
| ▶ | 1 | mujdatdurmaz... | 1 |
| | 2 | md@md.com | 1 |
| | 3 | mdd@mdd.com | 1 |
| * | NULL | NULL | NULL |

**Tablolar Arası İlişki Türleri**

**1'e 1 ilişki**

Örnek üzerinden konuyu anlatmaya çalışalım. Mesela bir bilet otomasyonu olsun. Bir tiyatro için bilet satılıyor ve her bileti yani koltuğu yalnızca bir kişi satın alabiliyor. Elimizde iki tane tablo bulunsun bunlardan biri müşteri tablosu ikincisi ise bilet tablosu. Bu iki tablo arasındaki ilişki birebir olur. Çünkü her müşteri yalnızca bir bilet satın alabiliyor ve her bilet yalnızca bir kişiye ait olabiliyor. Peki ilişkiyi kurarken hangi tabloya ekleme yapacağız? Müşteri tablosu mu Bilet tablosu mu? Bu durumu şu açıdan bakmamız gerekiyor. Eğer bilet tablosuna müşteri id'sini eklersem bu durum da satılmayan biletler için bu bölüm null değerini alacaktır. Ancak bu bizim istediğimiz bir durum değil. Onun yerine müşteri

tablosuna bilet id'sini eklersek her eklenen müşteri bir bilet satın almış olacağı için herhangi bir null değeri söz konusu değildir. Bu durumda örnek tablomuz aşağıdaki gibidir:

| id | musteri_adi | musteri_soyadi | adres | bilet_id |
|---|---|---|---|---|
| 1 | Selin | Demir | İstanbul | 1 |
| 2 | Ahmet | Kara | Ankara | 2 |
| 3 | Selim | Nadir | İstanbul | 3 |

| bilet_id | koltuk_no | fiyat |
|---|---|---|
| 1 | 23 | 20 |
| 2 | 12 | 35 |
| 3 | 45 | 15 |
| 4 | 54 | 15 |

**1'e Çok (1-n) İlişki**

En fazla rastalanan ilişki türüdür. Bir dershanede öğrenciler ve bu öğrencilere danışmanlık yapan öğretmenleri tutan iki adet tablomuz olsun. Bu iki tablo arasında kurulan ilişki 1'e n ilişki olur. Çünkü bir öğretmen birden fazla öğrenciye danışman olabiliyor ancak bir öğrenci en fazla bir öğretmenden danışmanlık alabiliyor. Peki hangi tabloya ekleme yaparak ilişkiyi kuracağız? Öğretmen tablosunu ele alalım. Eğer bu tabloya öğrenci id'lerini eklemeye başlarsak her öğrenci için o öğretmen tablosuna tekrar tekrar aynı öğretmenin verisini girmiş olacağız. Veri tekrarı en son isteyeceğimiz olaydır. Bu yüzden öğretmen id'lerini öğrenci tablosunda tutmak daha sağlıklı olacaktır. Şimdi doğru ve yanlış tablolara bakalım:

**1.Tablo Yanlış: Veri Tekrarı Var**

| id | ogretmen_adi | ogretmen_soyadi | ogrenci_id |
|---|---|---|---|
| 1 | Selin | Demir | 1 |
| 2 | Ahmet | Kara | 2 |
| 3 | Selim | Nadir | 3 |
| 2 | Ahmet | Kara | 4 |
| 2 | Ahmet | Kara | 5 |
| 3 | Selim | Nadir | 6 |

**2.Tablo Doğru: Veri Tekrarı Yok**

| id | ogrenci_adi | ogrenci_soyadi | ogretmen_id |
|---|---|---|---|
| 1 | Elif | Türkmen | 1 |
| 2 | Ayşe | Sarı | 1 |
| 3 | Ender | Kaya | 2 |
| 4 | Ali | Demir | 3 |
| 5 | Adem | Salih | 2 |

**Çoka Çok (n-m) İlişki**

Son ilişki türümüz en karmaşık olan çoktan çoğa ilişki türü. Bu ilişki türünde iki tabloda birden fazla bağa sahiptir. Bu yüzden iki tablo bu ilişkiyi açıklamak için yeterli olmaz. Yine bir ilişki türü örneği ile konuya giriş yapalım. Bir üniversitede ders seçimi yapan öğrenciler ile seçilen derslerin kayıtlarının tutulduğunu düşünelim. Bu durumda elimizde iki adet tablo bulunmaktadır:Öğrenci ve dersler. Bir öğrenci birden fazla ders seçebilirken, bir derste birden fazla öğrenci tarafından seçilebilmektedir. Bu durumda aralarında çoktan çoğa bir ilişki oluşmaktadır. Bu durumu tablolaştırırken bir üçüncü tabloya daha ihtiyacımız olmaktadır. Üçüncü tablo seçim tablosu olacak ve burada ders ile öğrencinin id'leri tutulacaktır. Aşağıdaki tabloda bu ilişkiyi görelim:

| id | ogrenci_adi | ogrenci_soyadi |
|---|---|---|
| 1 | Elif | Türkmen |
| 2 | Ayşe | Sarı |
| 3 | Ender | Kaya |
| 4 | Ali | Demir |
| 5 | Adem | Salih |

| id | ogrenci_id | ders_id |
|---|---|---|
| 1 | 1 | 3 |
| 2 | 1 | 5 |
| 3 | 2 | 1 |
| 4 | 3 | 4 |
| 5 | 4 | 2 |
| 6 | 4 | 3 |

| id | ders_adi |
|---|---|
| 1 | Matematik |
| 2 | Tarih |
| 3 | Edebiyat |
| 4 | Yazılım |
| 5 | İstatistik |

**DML VE DDL KOMUTLARI**

DML(Data Manupilation Language)

SELECT
INSERT
DELETE
UPDATE

Sorgularda genel olarak kullanıldı.

DDL(Data Definition Language)

CREATE DATABASE
CREATE TABLE
SELECT INTO
DROP
DELETE
ALTER

ÖRNEKLER:

```
CREATE DATABASE TEST1

CREATE TABLE Personel
(
id int NOT NULL  IDENTITY(1,1) PRIMARY KEY,
adi_Soyadi nvarchar(50),
Sehir nvarchar(30)
)

DROP TABLE Personel

SELECT * INTO Yedek FROM Personel

ALTER TABLE Personel ADD Yas int

ALTER TABLE Personel DROP COLUMN Yas

ALTER TABLE Personel ALTER COLUMN Ad varchar(40)
```