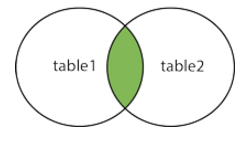
**SQL JOIN**

Şimdiye kadar aynı anda sadece tek bir tablodan veriler ile çalışıldı. Bazı durumlarda sonuç kümemizin daha anlamlı olabilmesi için iki farklı tablodan verilere ihtiyacımız olabilir. Bunun için sorgumuzda iki anahtar kelime daha kullanırız;

JOIN ifadesi ile hangi tabloları ve nasıl birleştireceğimizi, ON ifadesi ile tabloların hangi alanlar üzerinden birleşeceğini belirtiriz.

Tablolar genelde Primary Key(birincil anahtar) ve Foreign Key(yabancı anahtar) alanları üzerinden birleştirilseler de, gerektiğinde diğer herhangi bir alan da bunun için kullanılabilir. Fakat bu alanların aynı tür veri içerdiğinden emin olmalısınız.

**INNER JOIN**



SQL’de varsayılan (default) bağlantı türüdür. İlişkili tablolarda sadece JOIN koşulunu sağlayan kayıtlar listelenir.

**Örnek:**

Ürünlerimizin adlarını ve ait olduğu kategori adlarını gösteren sorguyu gösteriniz.

USE Northwind --Yazılan sorgunun hangi database için geçerli olacağını belirtiyorum

Select ProductName, CategoryName

from Products

inner join Categories on Categories.CategoryID = Products.CategoryID

**Örnek:**

Ürünlerin adlarını ve alındıkları toptancıların şirket adlarını gösteren sorguyu yazınız.

USE Northwind

Select ProductName, CompanyName

from Products

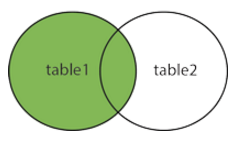
inner join Suppliers on Products.SupplierID = Suppliers.SupplierID

order by CompanyName

**OUTER JOIN**

LEFT veya RIGHT OUTER JOIN ifadeleri ilişkili alanda birbiriyle eşleşenlerin yanında eşleşmeyen kayıtları da listeler. JOIN koşuluna uymayan satırlar NULL(boş) değer olarak görüntülenirler.

**LEFT JOIN**



LEFT OUTER JOIN ile yazımdaki ilk tablonun tüm satırları listelenir ve diğer tablo ile eşleşmeyen alanlar NULL(boş) değer ile gösterilir. Eğer tabloların sorgudaki sırası değişirse aynı sonucu elde edebilmek için RIGHT OUTER JOIN kullanılır.

SELECT <sütun ad(lar)ı>

FROM <ilk tablo>

LEFT JOIN <ikinci tablo> ON <ilk tablo>.<anahtar alan>=<ikinci tablo>.<anahtar alan>

**Örnek:**

Aşağıdaki sorgunun SELECT kısmında iki sütun tek bir sütun adı altında birleştirilmiştir. Çalıştırıldığında çalışanın tam adı ve müşterilerden aldığı siparişler ve tarihleri listelenecektir.

USE Northwind

SELECT Employees.FirstName + ' ' + Employees.LastName AS Name, OrderID, OrderDate

FROM Employees

LEFT JOIN Orders ON Employees.EmployeeID = Orders.EmployeeID

**Örnek:**

Ürünleri ve bağlı bulundukları kategorileri listeleyiniz. Ancak urunu olmayan kategoriler de sorgu sonucuna dâhil edilsin.

Select ProductName, CategoryName

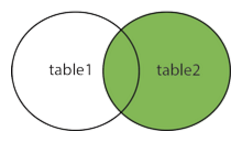
from Categories

left join Products on Categories.CategoryID = Products.CategoryID

--left outer join Products on Categories.CategoryID = Products.CategoryID

**Eğitmen Notu://** Left Join ile Left Outer Join arasında hiçbir fark yok. Execution Planda Left Join e dönüştürülüp öyle çalıştırılıyor.

**RIGHT JOIN**



İkinci tablodaki tüm kayıtları listelemek istendiğinde RIGHT JOIN kullanılır. Bu durumda ise birinci tablodaki eşleşmeyen kayıtlar NULL olarak görüntülenir. Eğer tabloların yazım önceliğini değiştirirseniz, sorgu LEFT OUTER JOIN ile aynı sonucu verecektir.

SELECT <sütun ad(lar)ı>

FROM <ilk tablo>

RIGHT JOIN <ikinci tablo> ON <birinci tablo>.<anahtar alan> = <ikinci tablo>.<anahtar alan>

**Örnek:**

Aşağıdaki örnekte tüm müşteriler ve verdikleri siparişlerin tarihleri listelenmiştir. Hiç siparişi olmayan müşterilerin tarih alanı NULL olarak görüntülendiğine dikkat edin.

USE Northwind

SELECT companyname, customers.CustomerID, orderdate

FROM orders

Right OUTER JOIN customers ON customers.CustomerID = orders.CustomerID

**Örnek:**

Ürünleri ve bağlı bulundukları kategorileri listeleyiniz. Ancak kategorisi olmayan ürünler de sorgu sonucuna dâhil edilsin.

Select ProductName, CategoryName

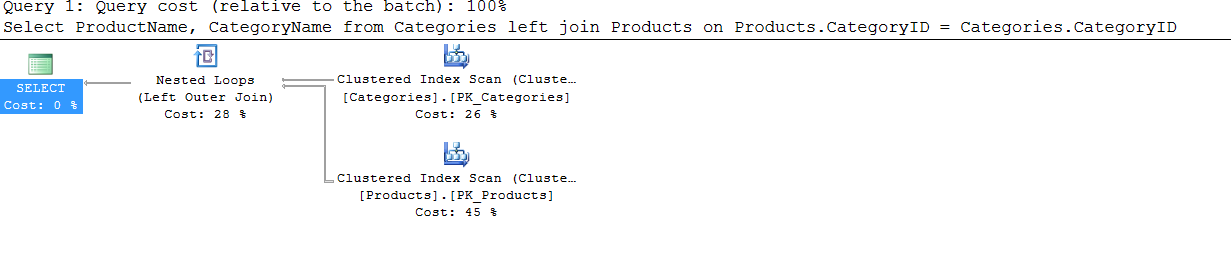
from Categories

right join Products on Products.CategoryID = Categories.CategoryID

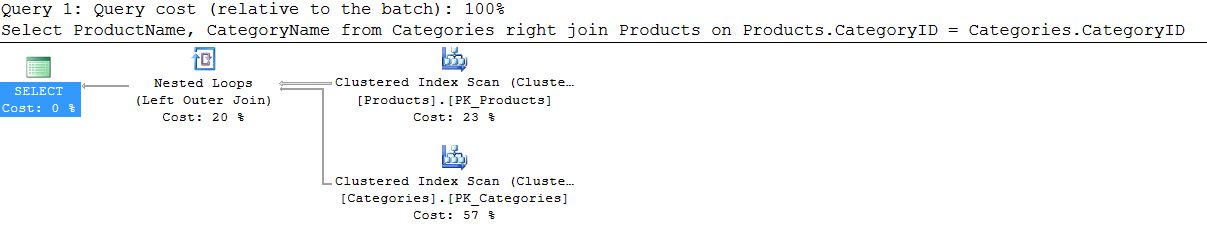
-- right outer join Products on Products.CategoryID = Categories.CategoryID

**Eğitmen Notu://** Right Join ile Right Outer Join arasında hiçbir fark yok. Execution Planda Right Join e dönüştürülüp öyle çalıştırılıyor.

**Eğitmen Notu://** Right Join execuion planda Left Join e dönüştürülerek işlem görür. Aşağıda gördüğün ilk resim left join ile yazılan bir sorgunun execution planda ki işleme şeklidir.

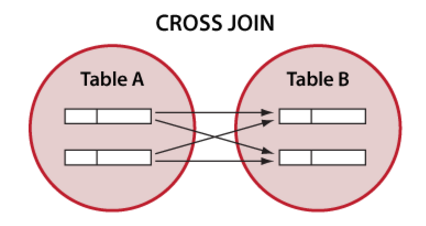


Aşağıdaki resimde de right join ile yazılan bir sorgunun execution plandaki işleme şeklidir.



Her iki resimde de dikkat ederseniz sorgu right join ile de yazılmış olsa işlenmesi left join yapısıyla olur.

**CROSS JOIN (Gelebilecek Sorulara Karşı Eğitmen Bilmeli)**



İlişkili tablolar arasında olası tüm eşleşmeleri listeler. Belirli ortak bir alan belirtilmesine gerek yoktur.

**Örnek:**

Aşağıdaki örnek üreticilerin çalışabilecekleri tüm olası gemi şirketlerini görmemizi sağlar. Shippers tablosu 3 kayıt ve Suppliers tablosunda 29 kayıt olduğundan, sonuç kümesinde 87 satır görüntülenecektir.

USE Northwind

SELECT suppliers.companyname, shippers.companyname

FROM suppliers CROSS JOIN shippers

**Not://** Cross join in “join” anahtar kelmesini eklemeden diğer yazım şekilleri (Bu yazım şekilleri ikiden fazla tabloları birleştirmek istediğimizde karışık bir yapı olduğu için kullanılması pek tavsiye edilmez. Ayrıca “Where” den sonra koşulda belirtebilirsiniz ama cross join mantığında çalışmaktadır.):

1)

USE Northwind

SELECT suppliers.companyname, shippers.companyname

FROM suppliers, shippers

2)

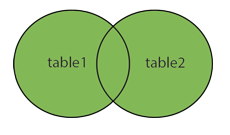
USE Northwind

SELECT suppliers.companyname, shippers.companyname

FROM suppliers, shippers

where 1=1

**FULL JOIN**



Her iki tablodaki tüm verileri getirmek için kullanılır.

**Örnek:**

Ürünleri ve bağlı bulundukları kategorileri listeleyiniz. Ancak kategorisi olmayan ürünler ve ürünleri olmayan kategoriler de sorgu sonucuna dâhil edilsin.

Select ProductName, CategoryName

from Categories

full join Products on Products.CategoryID = Categories.CategoryID

--full outer join Products on Products.CategoryID = Categories.CategoryID

**Not://** Full Join ile Full Outer Join arasında hiçbir fark yok. Execution Planda Full Join e dönüştürülüp öyle çalıştırılıyor.

**Not://** İki tablo arasında NULL olanları listelemek istediğimizde “IS NULL” parametresi kullanılır.



**İkiden Fazla Tablonun Birleştirilmesi**

Gerektiğinde ikiden fazla tabloyu birleştirmek de mümkündür. Bir JOIN işlemiyle birleştirilen tablolar bir başka tabloya aralarındaki ortak bir sütun üzerinden bağlanabilir.

**SELF JOIN**

Bazen, tek bir tablo üzerinde JOIN işlemleri gerçekleştirmeye ihtiyaç duyarız. Bu tip JOIN çeşidine SELF JOIN denilmektedir.

**Örnek:**

Hangi çalışanın kime rapor verdiğini rapolayanız. Rapor veren ve rapor verilen olarak.

select emp.FirstName + ' ' + emp.LastName as RaporVeren , e.FirstName + ' ' + e.LastName as RaporVerilen

from Employees e

inner join Employees emp on e.EmployeeID=emp.ReportsTo

**UNION**

UNION fonksiyonu, iki veya daha fazla SELECT sorgusunun sonuçlarını tekbir sonuç kümesinde birleştirir. Görüntülenecek olan tablolar aynı veri türünde aynı sayıda ve aynı düzendeki sütunlardan oluşmalıdır. UNION kullanırken dikkat edilecek nokta listelenecek kolonların aynı türde, birbiri ile uygun türde veri tiplerine sahip olmasıdır.

**Örnek:**

Aşağıdaki örnekte Customers tablosu ve Employees tablosu verileri birleştirilmiştir. Birinci sorgunun SELECT ifadesindeki takma adın görüntülendiğine dikkat ediniz.

SELECT (firstname + ' ' + lastname) AS Name, city, postalcode

FROM Employees

UNION

SELECT companyname, city, postalcode

FROM Customers

**Örnek:**

SELECT CompanyName, Phone, ContactName FROM Customers

UNION

SELECT CompanyName, Phone, ContactName FROM Suppliers

UNION

SELECT CompanyName, Phone, 'Kontak Yok' FROM Shippers

**Not://** Sütun sayısı ve cinsleri aynı olmak zorunda. 'Kontak Yok' bu yüzden yazıldı.

**UNION ALL**

Union benzer satırları atar, yani inner join görevi yapar. Fakat tüm satırların hepsini görmek istiyorsak yani full join istiyorsak union all kullanmalıyız.

SELECT CategoryID, CategoryName FROM Categories

UNION All

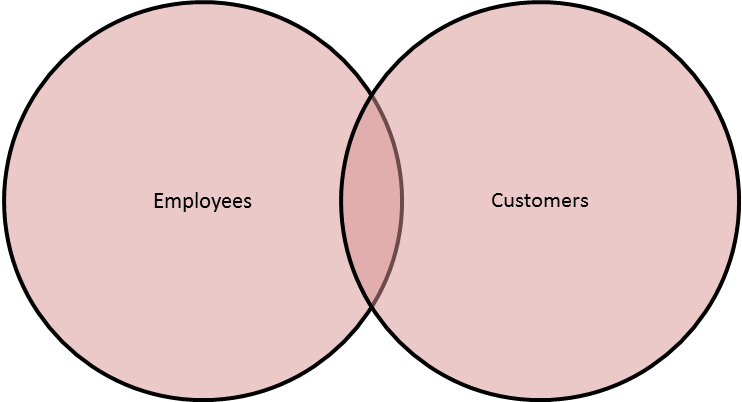
SELECT ShipperID, CompanyName FROM Shippers

**Union ve Union All Farkı (Eğitmen için...)**

**UNION ALL** bütün sonuçları herhangi bir filtreleme yapmadan getirir. **UNION** ise, **UNION ALL** ile gelen bütün kayıtları bir tablo olarak düşünürsek, bu tablo üzerinde **DISTINCT** ile bir select alıyormuşuz gibi sonuç getirmektedir. Yani tekrarlayan satırları getirmemektedir. Bu iki farkı daha iyi anlamak için bir örnek ile açıklayalım.

**INTERSECT ( Eğitmenin Bilmesi gerek, gelebilecek sorulara karşı)**

Adından da anlaşıldığı üzere bir kesişim ifadesinden kalan sonucu bize verecektir. INTERSECT komutu bizim farklı iki sql sorgumuzdan çıkacak olan aynı sonuçları bizim önümüze getirmemize yaramaktadır. Yukarıdaki resimden de anlaşıldığı üzere iki farklı sorguda ki aynı çıkan sonuçları göstermektedir. Yani beyaz olan kısımdır. Inner join ile aynı mantıktadır. Sadece tablo arasında değil, iki farklı sorgu için kullanılır.



**INTERSECT**

**EXCEPT** **(Eğitmenin Bilmesi gerek, gelebilecek sorulara karşı)**

İlk sorguda olup ikinci sorguda olmayan verileri getirir.

**Eğitmen Notu://** Union veya diğer ifadelerle oluşturulmuş result set başka bir table a insert edilmek isteniyorsa sadece ilk table ın önüne **INSERT INTO**ifadesi yazılır. Diğer table larında önüne yazılırsa query hata verecektir.

Bu işlemi denemek için ilk önce bir table create edelim. Insert işlemini bu table ın içine yapacağız.

Şimdi union işlemi yaparak CustTable3 e insert yapalım.

INSERT INTO CustTable3

SELECT Name,SurName from CustTable1

UNION

SELECT Name,SurName from CustTable2

CustTable3 ün sonucuna bakalım.

|  |
| --- |
| SELECT \* FROM CustTable3 |

[](http://www.turgaysahtiyan.com/photo/SQLServerdaUNIONUNIONALLEXCEPTveINTERSEC_E1AA/image_7.png)

**Order By Kullanımı**

Union ve yukarıda bahsettiğimiz diğer ifadelerde order kullanımında Order ifadesi her bir sorgu için yazılmaz. Order ifadesi en sona konulur ve en son birleştirilmiş sonuç için order edilmesi sağlanır.

SELECT Name,SurName from CustTable1

UNION ALL

SELECT Name,SurName from CustTable2

ORDER by Name ASC,SurName DESC