***SQL***

**SQL Nedir?**

SQL (Structured Query Language)

SQL Yapılandırılmış sorgu dili anlamına gelir.

Sorgu dili, veri tabanlarından belirli bilgilerin alınmasını kolaylaştırmak için tasarlanmış bir tür programlama dilidir.

Bu nedenle SQL; "veri tabanlarının dili" olarak adlandırlır.

**Yeni Tablo Oluşturma**

Tables > Sağ click > New tables

Column Name : Sütün ismi

Data Type : Tipi

Allow Nulls : Boş olmasına izin ver

Not: Veri tabanı bağlantısı koptuğunda vs.

use tablo adı

**Bazı Data Tipleri**

**int :** Tam sayı

**char(10) :** Karakter (verilen sayı kadar karakter girilebilir, harf de olabilir rakam da olabilir nokta virgül olabilir)

**varchar(50) :** Maksimum 50 minumum 1 karakter girebilirim. **Not:** varchar'ın char dan farkı diskte 50 karakterlik yer kaplamıyor örneğin varcharda ali yazdık sadece 3 byte lık alanı dolduruyor.

**nchar :** Örneğin arapça gibi çince gibi farklı dillerde kayıt tutabilmeye yarar.

**nvarchar:** Verinin boyutu kadar yer kaplar. Varchardan farklı olarak unicode’u destekler.4000 karaktere kadar değer alabilir.

**date :** Gün / Ay / Yıl

**datetime :** Gün / Ay / Yıl / Saat / Saniye

**bit :** Örneğin erkek mi sorusuna 1 veya 0 cevabını veririz 1 ise erkek 0 ise kız ve 1 bitlik yer kaplar.

**tinyit :** 0 ile 255 arasındaki sayıları tutmak için kullanılır. Veri tabanında 1 byte’lık yer kaplar.

**text:** Belirtilenden az değer girilse bile boyutu kadar yer kaplar.2GB’a kadar metinsel veri depolar. Unicode’u desteklemez. **Unicode:** Farklı dillere özgü karakterlerdir.

**ntext:** Text’den farklı olarak girilen karakter boyutu kadar yer kaplar ve unicode’u destekler.

**bigint:** 8 byte büyüklüğünde -2⁶³ ve 2⁶³ arasında değer tutabilen tam sayı veri tipidir.

**smallint:** 2 byte büyüklüğünde -32.768 ve 32.768 arası değer alabilen tam sayı veri tipidir.

**time:** Sadece saat verilerini saat-dakika-saniye-salise(7 basamaklı) şeklinde saklayan veri tipidir.

**SQL Query (Sorgu)**

SQL'de bazı işlemleri gerçekleştirmesi için bazı komutlar verilir.

Bu komut veya talimatlara query (sorgu) denir.

Sistem bu sorguyu girdi olarak alacak ve istenen çıktıyı verecektir.

**Tablo ile ilgili komutlar;**

CREATE TABLE (Yeni Tablo Oluşturma)

ALTER TABLE (Tabloyu Günceller)

DROP TABLE (Tabloyu Siler)

New Query ile yeni bir query ekranı açılır.

**CREATE Komutu;**

CREATE TABLE tablo ismi (

OgrenciNo int NOT NULL,

Ad varchar(30) NOT NULL,

Soyad varchar(30) NOT NULL,

TCKimlikNo char(11) NOT NULL,

KayitTarihi date NOT NULL)

ardından "Parse" edilir ve kontrol edilir ardından "Execute" Edilir ve yeni bir tablo oluşturulur.

Not: NOT NULL komutu yazarsak boş bırakılmamasına izin verilmez.

**ALTER Komutu;**

ALTER TABLE tablo\_ismi ADD sütun\_ismi datatype (tabloya yeni bir sütun ekler)

ALTER TABLE tablo\_ismi ALTER COLUMN sütun\_ismi datatype (var olan sütunu günceller)

Örneğin;

ALTER TABLE Ogrenci ADD DogumTarihi date

Allow Nulls tikli olmaması için; ALTER TABLE tablo\_ismi ADD sütun\_ismi datatype NOT NULL

ALTER TABLE tablo\_ismi ALTER COLUMN sütun\_ismi datatype NOT NULL

**DROP Komutu;**

DROP TABLE tablo adı yazılır ardından "Execute" edilir ve belirlenen tablo silinir.

**SELECT ve WHERE Komutları;**

New Query ile yeni bir query ekranı açılır.

**SELECT Komutu;**

SELECT \* FROM tablo ismi (Tablodaki tüm sütunları çağırır)

SELECT Ad, Soyad FROM tablo ismi (Tablodaki "Ad ve Soyad" sütunlarını çağırır)

ardından "Execute" edilir

**WHERE Komutu;**

SELECT \* FROM Ogrenci WHERE ErkekMi = 1

Yani Ogrenci tablosundan ErkekMi sütunundan erkek olanları yani true 1 olanları çağır anlamına gelir.

ardından "Execute" edilir

String olarak çağırırken ise tek tırnak içinde yazmalıyız.

SELECT \* FROM Ogrenci WHERE Ad = 'Eren'

Tarih olarak çağırırken ise tek tırnak içinde yazarız fakat belirlenen tarihten büyük mü, küçük mü, eşit mi belirlemek için <,>,= kullanılır.

SELECT \* FROM Ogrenci WHERE KayitTarihi >= '01.15.2020'

01.15.2020 tarihini ve sonrasında ki verileri getirir öncesini getirmez.

Not: Tarihler yazılırken önce ayı sonra günü sonra yılı yazmamız gerekiyor ( AY / GÜN / YIL ) '01.15.2020'

Ya da ( YIL / AY / GÜN ) olarak da yazabiliriz ama arada nokta olmadan '20201501'

**Operatörler**

AND

OR

in

like

**AND Komutu;**

SELECT \* FROM Ogrenci WHERE OgrenciNo > 500 AND ErkekMi = 1

**OR Komutu;**

SELECT \* FROM Ogrenci WHERE Soyad = 'Doğan' OR Ad = 'Eren'

**in Komutu;**

SELECT \* FROM Ogrenci WHERE OgrenciNo in (725, 954, 1204)

**like Komutu;**

SELECT \* FROM Ogrenci WHERE Ad like 'E%'

Adının baş harfi E olan öğrencileri getirir.

SELECT \* FROM Ogrenci WHERE TCKimlikNo like '%34%'

3 ve 4 başta da olabilir sonda da olabilir 3 ve 4 yanyana olan tc kimlik numaralarını getirir.

**Not:** SQL de yorum satırı yapmak için kodun başına iki adet slash konur –

**insert (INSERT) Komutu**

insert into tablo\_adı values(925, 'Emre', 'Erkan', '12345678905', '2010-05-19', 1, '19821223', 70,95,0)

Ogrenci tablosundaki tüm sütunları doldur anlamına gelir.

Parantez içine tablodaki sütunları sırayla dolduruyoruz.

Bu kodda (OğrenciNo, Ad, Soyad, TCNo, KayİtTarihi, ErkekMi, DogumTarihi, SınavNotu1, SınavNotu2, FinalNotu)

verilerini yazdırır.

Ayrıca insert komutunda tablodaki bazı sütunlar allow nulls tikli ise yani boş bırakılabilir ise insert komutunu yazarken onları yazmasakta yeni bir satır eklemesi yapar.

insert into Ogrenci (OğrenciNo, Ad, Soyad, TCNo, KayİtTarihi, ErkekMi, DogumTarihi)

values(980,'Damla', 'Acar', '54297541264', '2021-08-20', 0, '20040729')

Ogrenci tablosundaki şu sütunları doldur anlamına gelir.

Sınav1-2 ve final notları allow nulls tikli olduğu için yazmasakta olur.

Allow nulls tikli değil ise ve bir tane sütuna veri eklemezsek kod hata verir yeni satırı eklemez.

**UPDATE Komutu**

UPDATE komutu veri tabanındaki tablomuzda bulunan kayıtları güncellemeye yarar.

Tek bir satırda güncelleyebiliriz, daha fazla satırı da güncelleyebiliriz veya hepsini güncelleyebiliriz.

update Ogrenci set SinavNotu1 = 75 where Ad = 'Betül' and Soyad = 'Doğan'

Aynı anda 2 tane veriyi değiştirmek için;

update Ogrenci set SınavNotu2 = 85, FinalNotu = 87 where ErkekMi = 0 and OgrenciNo >= 900

**delete komutu**

Bir ya da birden fazla satırı silmeye yarar.

Tüm tabloyu silmek için;

delete from Ogrenci

Bir tane sütun silmek için;

delete from Ogrenci where OgrenciNo = 901

Birden fazla sütun silmek için;

delete from Ogrenci where KayıtTarihi > '2022-09-14'

**order by komutu;**

Verileri artan ya da azalan düzende sıralamak için kullanılır.

select \* from Ogrenci order by OgrenciNo

Öğrenci numaralarını sıralar.

**asc komutu;**

select \* from Ogrenci order by OgrenciNo asc

Öğrenci numaralarını artarak sıralar (Küçükten büyüğe doğru sıralar)

**desc komutu;**

select \* from Ogrenci order by OgrenciNo desc

Öğrenci numaralarını azalarak sıralar (Büyükten küçüğe doğru sıralar)

**Not:** 2 adet koşul da verebiliriz.

Örneğin cinsiyete göre sıralayıp onuda isimlere göre sıralayabilir.

select \* from Ogrenci order by ErkekMi, Ad

**between komutu;**

Bir veri listesinde belli bir alan aralığı ve ya değer aralığında ki verileri listelemek istediğimiz zamanlarda kullanırız.

select \* from Ogrenci where OgrenciNo between 600 and 800

Öğrenci numarası 600'e büyük eşit ve 800'e küçük eşit olan öğrencileri listeler.

select \* from Ogrenci where DogumTarihi between '1999-01-01' and '2003-01-01'

Doğum tarihleri bu aralıkta olanları sıralar.

**Not:** Aynı yılda doğanları çağırmak için;

select \* from Ogrenci where DogumTarihi between '1999-01-01' and '1999-12-31'

**count komutu;**

Sorgu tarafından döndürülen kayıtların sayısını hesaplar.

select count(\*) from Ogrenci

Ogrenci tablosunda kaç tane kayıt olduğunu gösterir.

select count(\*) from Ogrenci where ErkekMi = 1

Erkek öğrencilerin sayısını gösterir.

**sum komutu;**

Seçilen sütunun değerlerinin toplamını verir.

select sum(FinalNotu) from Ogrenci

FinalNotu sütunun toplamını gösterir.

**max komutu;**

Seçilen sütunun en büyük değeri çağırır.

select max(FinalNotu) from Ogrenci

FinalNotu sütunundaki en yüksek puanı gösterir.

**avg komutu;**

İstenen sütunun değerlerini toplar ve ortalamasını verir.

select avg(FinalNotu) from Ogrenci

FinalNotu sütunundaki puanları toplar ve ortalamasını gösterir.

select sum(FinalNotu) / count(\*) from Ogrenci

avg komutu kullanmadan da ortalama hesaplayabiliriz.

**group by komutu;**

Tablodaki belli sütunlardaki verileri gruplandırıp sorgu oluşturmaya yarar.

select ErkekMi, count(\*) as 'Öğrenci Sayısı' from Ogrenci group by ErkekMi

Sonuç;

|  |  |  |
| --- | --- | --- |
|  | ErkekMi | Öğrenci Sayısı |
| 1 | 0 | 3 |
| 2 | 1 | 4 |

**Not:** as komutu kullanarak sütun ismi belirtebiliyoruz.

**Not:** Gruplandırmak istediğimiz sütunun ismini select' ten sonra yazmamız gerekir.

Yazmazsak sonuç tablosunda ErkekMi kısmı olmaz sadece Öğrenci Sayısı kısmı olur.

select ErkekMi, count(\*) as 'Öğrenci Sayısı',

avg(SinavNotu1) as 'Sınav1 Ort',

avg(SinavNotu2) as 'Sınav2 Ort',

avg(FinalNotu) as 'Final Ort',

max(FinalNotu) as 'En Yüksek Final Notu'

from Ogrenci group by ErkekMi

Sonuç;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ErkekMi | Öğrenci Sayısı | Sınav1 Ort | Sınav2 Ort | Final Ort | En Yüksek Final Notu |
| 1 | 0 | 3 | 28 | 28 | 29 | 87 |
| 2 | 1 | 4 | 35 | 90 | 23 | 95 |

**primary key komutu;**

Bir tablodaki kayıtları birbirinden ayırt etmemizi sağlayan bir kısıtlama (constraint) yapısıdır.

O tabloda bulunan veri sıralarına ait bir "benzersiz tanımlayıcıdır".

Benzersiz (Unique) olmalıdır. NULL değerine sahip olamaz.

Örneğin öğrenci numarası ve ya TC kimlik numarasından öğrencileri birbirinden ayırabiliriz.

Bir sütun nasıl primary key yapılır;

**Tabloya sağ click ile tıkla > Design kısmına tıkla > primary key yapmak istediğimiz sütuna sağ click ile tıkla > Remove primary key'e bas**

**Birden fazla tablodaki verileri birleştirmek için;**

**select tablo adı.tablodaki sütun, tablo adı.tablodaki sütun, ...**

**from tabloadı1,tabloadı2**

**where tabloadı1.çağırdığımız sütun = tabloadı2.çağırdığımız sütun**

**(Çağırdığımız sütundaki birbirine eşit olan verileri getirir)**

**select Ogrenci.OgrenciNo, Ogrenci.Ad, Ogrenci.Soyad, Ogrenci.KayitTarihi,**

**OgrenciAdres.AdresDetay, OgrenciAdres.Sehir from Ogrenci,OgrenciAdres**

**where Ogrenci.OgrenciNo = OgrenciAdres.OgrenciNo**

**JOİNLER;**

İki ya da daha fazla tabloyu aynı anda sorgulayarak bir sonuç tablosu (result table) oluşturmaya yarar.

**inner join (iç birleştirici) komutu;**

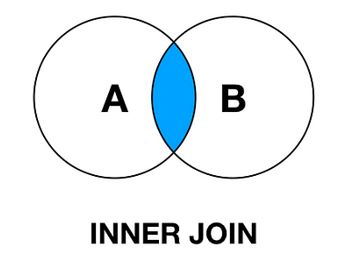
İki ya da daha fazla tabloda ortak olan iki alandaki değerleri kontrol ederek tabloları birleştirir.

**select** A.OgrenciNo, A.Ad, A.Soyad, A.KayitTarihi, B.AdresDetay, B.Sehir from Ogrenci A

inner join OgrenciAdres B

on A.OgrenciNo = B.OgrenciNo

Ogrenci tablosunu 'A' olarak, OgrenciAdres tablosunu 'B' olarak kısalttık ve tablodaki sütunları çağırırken tablo ismi yazmak yerine kısaltış halde yazarak çağırdık.



inner joindeki verileri sıralamak için ve toplam tutarı görmek için;

**select A.OgrenciNo, A.Ad, A.Soyad, SUM(B.MiktarTL) as 'Toplam Harcama' from Ogrenci A**

**inner join KantinKayıtlari B**

**on A.OgrenciNo = B.OgrenciNo**

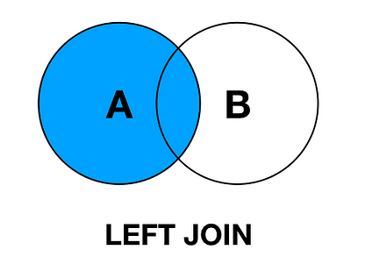
**group by A.OgrenciNo, A.Ad, A.Soyad**

**order by OgrenciNo**

**Left Outer (dış birleştirici) Join;**

Outer Join (dış birleştirici)

Birincil tablodaki tüm kayıtları ve eşleşen kayıtları olan diğer tablodaki kayıtları birleştirir.



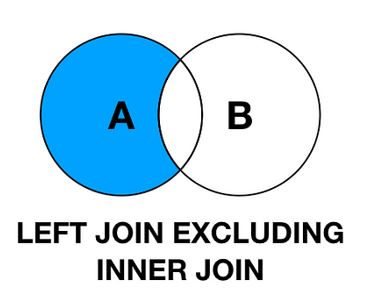
select A.OgrenciNo, A.Ad, A.Soyad, B.AdresDetay, B.Sehir from Ogrenci A

left join OgrenciAdres B

on A.OgrenciNo = B.OgrenciNo

İster left join ister left outer join yazabiliriz ikiside aynı şey olduğu için her türlü çalışır.

Sadece A'nın içindekileri çağırmak için;

select A.OgrenciNo, A.Ad, A.Soyad, B.AdresDetay, B.Sehir from Ogrenci A

left join OgrenciAdres B

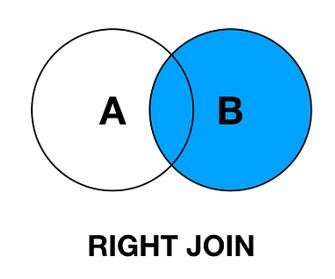
on A.OgrenciNo = B.OgrenciNo

where B.OgrenciNo is null

A tablosunda olup B tablosunda olmayan öğrencileri getirir.

**Right Outer Join;**

İkincil tablodaki tüm kayıtları ve eşleşen kayıtları olan diğer tablodaki kayıtları birleştirir.

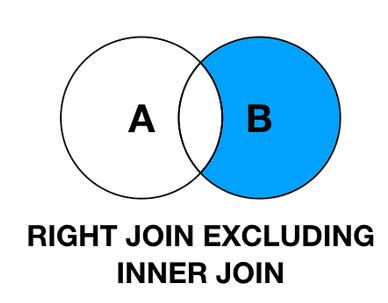


select A.OgrenciNo, A.Ad, A.Soyad, B.AdresDetay, B.Sehir from Ogrenci A

right join OgrenciAdres B

on A.OgrenciNo = B.OgrenciNo

Sadece B'nın içindekileri çağırmak için;

select A.OgrenciNo, A.Ad, A.Soyad, B.AdresDetay, B.Sehir from Ogrenci A

right join OgrenciAdres B

on A.OgrenciNo = B.OgrenciNo

where A.OgrenciNo is null

**Full Outer Join;**

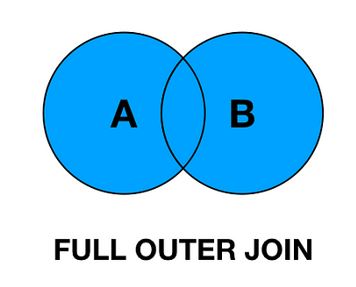
İki tablodaki eşleşen kayıtlar ve eşleşmeyen sol ve sağ kayıtlar için kullanılır.

A tablosunda olup B tablosunda olmayan verileri null olarak gösterecek aynı şekilde B tablosunda olup A tablosunda olmayan verileri null olarak gösterecek.

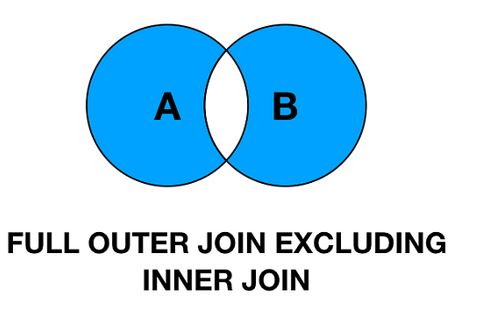
select A.OgrenciNo, A.Ad, A.Soyad, B.AdresDetay, B.Sehir from Ogrenci A

full join OgrenciAdres B

on A.OgrenciNo = B.OgrenciNo



İki tablodaki ortak veriler gelmemesi için sadece tabloların kendi verilerinin gelmesi için;

select A.OgrenciNo, A.Ad, A.Soyad, B.AdresDetay, B.Sehir from Ogrenci A

full join OgrenciAdres B

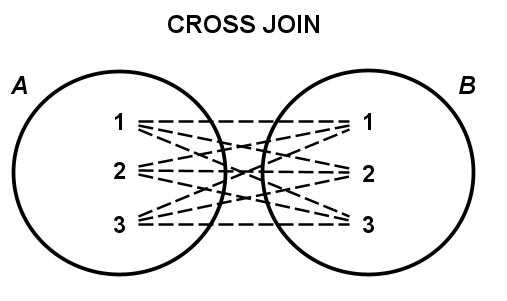
on A.OgrenciNo = B.OgrenciNo

where A.OgrenciNo is null

or B.OgrenciNo is null

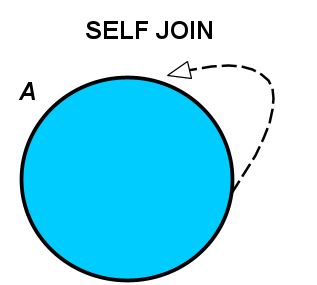
**Cross Join (çapraz birleştirici);**

Tablolar arasında yapılan birleştirme işleminde seçilen alanlar arasındaki tüm kombinasyonları sonuç tablosu olarak vermeyi sağlar.



**Self join;**

Bir tablonun kendisiyle JOIN işlemine tutulmasına Self Join denir. Hiyerarşik verileri sorgulamak veya aynı tablodaki satırları karşılaştırmak için kullanışlıdır.



**Alt Sorgular;**

Alt sorgular SQL içerisinde kullanılan SELECT sorgusunun özel bir türüdür. Veri tabanı tasarımı sırasında verilerin tekrarını önlemek amacıyla nesnelere parçalara ayrılır.

select \* from Ogrenci where OgrenciNo in (select OgrenciNo from OgrenciAdres where Sehir = 34)

Ogrenci tablosundan OgrenciAdres tablosunda kayıtlı olan 34(İstanbullu) olan öğrencileri getirir.

**Not:** in yerine = kullanırsak bir kişi varsa bir problem olmaz fakat birden fazla ise hata verir in kullanmamız lazım.

select \* from Ogrenci where OgrenciNo in

(select OgrenciNo from KantinKayıtlari where MiktarTL >=30)

Ogrenci tablosundan KantinKayıtlari tablosundan kayıtlı olan 30 TL ve üstü para harcayan öğrencileri getirir.