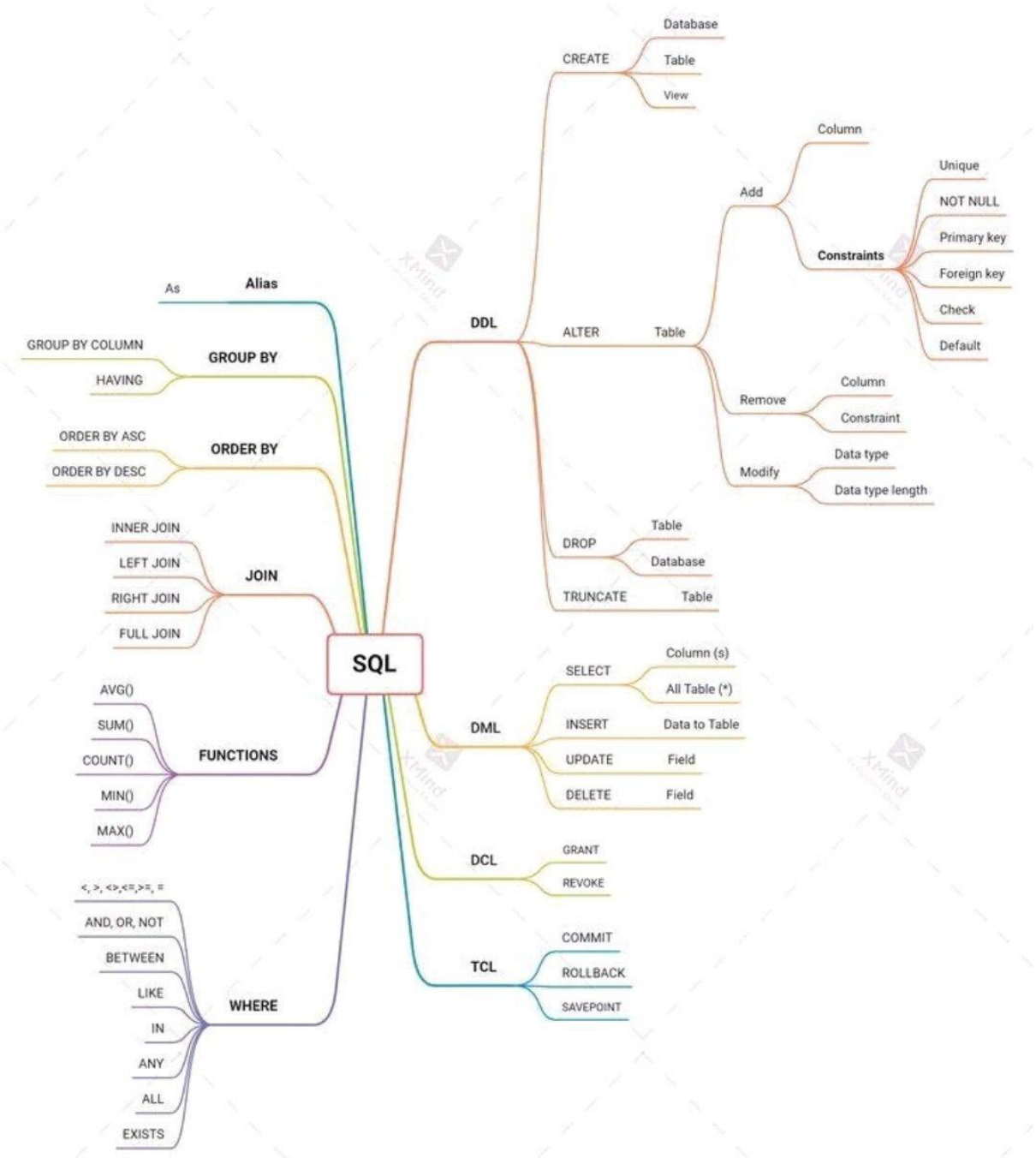
İÇİNDEKİLER

SQL - T-SQL

Kelime açılımı, “Structered Query Language” SQL, Türkçe ’ye “yapısal sorgulama dili” olarak geçmiştir. Temel olarak SQL’e bir sorgulama dili demek mümkün. Veri tabanı işlemlerini hızlı bir şekilde yapmak için kullanılan bir alt dil diye ifade edebiliriz. SQL ifadesi bir nevi veri tabanı yönetimi kavramını ifade etmektir.

T-SQL : SQL, düzeltilmesi veya değiştirilmesi istenen bilgileri açıkça belirtmeye izin veren ve yerine getirilebilecek başlıca işlemleri tanımlamamızı sağlayan bir komut takımıdır. Bu komutların oluşturduğu yapıya T-SQL dili denir. T-SQL ile veri ve sorgulara erişebilir, güncelleyebilir ve ilişkisel veri tabanı sistemi yönetilebilir. T-SQL komutları kullanım amaçlarına göre üç genel kategoriye ayrılır.



SELECT KOMUTU

SQL de veri seçme işlemi sağlamaktadır. SELECT AGE FROM CUSTOMER

ALİAS KULLANIMI

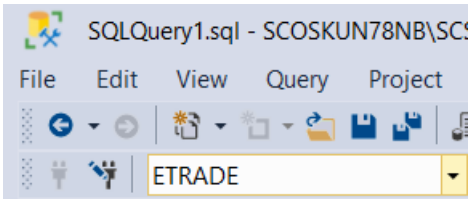
Takma isim oluşturmak için kullanılır tablo başlığını yazdıktan sonra AS ya da BOŞLUK 'tan sonra geçici olarak tanımlayacağımız tablo başlığı yazılır

```
SELECT NAMESURNAME, AGE AS YAS FROM CUSTOMER
```

	NAMESURNAME	YAS
1	Fikriye OYGUR	66

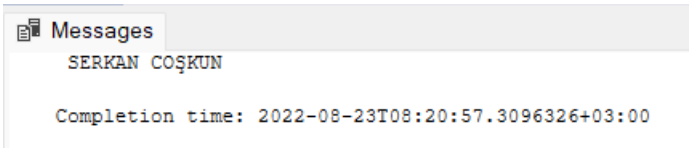
USE KOMUTU

Kullanacağımız veri tabanını çağırmamızı sağlar USE ETRADE



PRINT KOMUTU

Messages kısmında sonuç getirir. PRINT 'SERKAN COŞKUN'



CONVERT - CAST METODU

Geçici olarak değişkenin data type'ını değiştirmeyi sağlar. Fiziksel olarak etkilemez.

ÖR: İnt tipindeki bir veriyi nvarchar veri ile yan yana yazmak istiyorum fakat int ile nvarchar data typelarından dolayı yan yana gelememektedir. Bunun için int tipindeki verimi sadece o sorgumda olmak şartıyla geçici olarak data type'ı değiştiririz.

İki çeşit yöntem vardır biri convert diğeri cast metodu.

```
SELECT NAMESURNAME + CONVERT(NVARCHAR(MAX),AGE) FROM CUSTOMER

SELECT NAMESURNAME + CAST (AGE AS NVARCHAR) FROM CUSTOMER
```

WHERE KOMUTU

Select sorgusunu şarta veya şartlara bağlamamızı sağlar. Select sorgusundan sonra yazılır.

SELECT * FROM CUSTOMER WHERE NAMESURNAME...

- Where şartında "%" operatörü ile metinsel aramalarda % işaretinin yerine göre kelimenin başlangıcı bitişi ya da kelimenin içinde aramamızı sağlayabilmektedir.
- WHERE NAMESURNAME > 'C%' İLK HARFİ C DEN ALFABETİK SIRA İLE BÜYÜK OLANLARI GETİRİR.
- WHERE NAMESURNAME LIKE N'SERKA%' -> N HARFİ TÜM DİLLERE ALGILA ANLAMINA GELİR. STRİNG SORGULAMALARINDA YA DA STRİNG İÇİNDE KARAKTER ARAMALARINDA **LIKE** OPERATÖRÜ KULLANILIR.
- WHERE AGE BETWEEN 18 AND 40 -> BETWEEN ARASINDA ANLAMINA GELMEKTEDİR. YAŞI 18 İLE 40 ARASINDA OLAN DEĞERLERİ GETİRİR.
- WHERE AGE " (=)"Age'e eşit değerleri getirir. "(>)"Age'den küçük olan değerleri getirir. " (<)" Age'den küçük olan değerleri getirir"
  Eşit değildir için (<>) YA DA (!=) kullanırız.
  ÖR: WHERE AGE > 40 → AGE DEĞERİ 40DAN BÜYÜK OLAN DEĞERLERİ GETİRİR.
- WHERE NAMESURNAME LIKE N'R%T' ->İSMİNİN İLK HARFİ R SON HARFİ T
- WHERE CITY = 'TOKYO' AND AGE > 18 -> ŞEHRİ TOKYO VE YAŞI 18 DEN BÜYÜK OLAN DEĞERLERİ GETİRİR.
- WHERE CITY = 'TOKYO' OR AGE > 18 → **ŞEHRİ** TOKYO YA DA YAŞI 18 DEN BÜYÜK OLAN DEĞERLERİ GETİRİR.
- WHERE ADI LIKE '[NMR]%' → İSMİNİN İLK HARFLERİ N,M YA DA R İLE BAŞLAYAN DEĞERLERİ GETİRİR. [ ] İÇİNE ALINAN DEĞERLERİ AYRI AYRI SORGULAYARAK GETİRMEKTEDİR.
- WHERE ADI LIKE '[A-F]%' → ALFABETİK OLARAK A DAN F YE ANLAMINA GELİR
- WHERE ADI LIKE '[^A]%' → " ^ " İŞARETİ İLE BAŞ HARFİ A OLMAYAN DEĞERLER GETİRİLİR.
- WHERE NAMESURNAME LIKE '[H]%AN%' WHERE İLE BAŞ HARFİ H OLUP İÇİNDE AN OLANLARI BUL
- WHERE SEHIR IN ('LONDON','TOKYO') → OR ANLAMINA GELİR.
  İÇİNDE LONDON YA DA TOKYO OLAN ŞEHİRLERİ GETİR ANLAMINA GELİR.
- WHERE CUSTOMERNAME IS NOT NULL(NULL OLMAYANLARI GETİR)
- WHERE AGE NOT BETWEEN 25 AND 45 → NOT BETWEEN İLE ARASINDA OLMAYAN DEĞERLERİ GETİRİR.
- WHERE YEAR(CREATEDDATE)=2015 → CREATEDDATE YILI 2015 OLAN DEĞERLERİ GETİRİR.
- WHERE DAY (CREATEDDATE)=29 → CREATEDDATE GÜNÜ 29 OLAN DEĞERLERİ GETİRİR.

AGGREGATE FONKSİYONLARI

T-SQL sorgularımızda bir dizi değer üzerinde hesaplama yaparak sonuç değere ulaştıran fonksiyonlara Aggregate fonksiyonları denir.

Aggregate Fonksiyonlar:

- AVG(ORTALAMA DEĞERLERİ GETİRİR.)

  SELECT AVG(AGE) FROM CUSTOMER =41 DEĞERİNİ GETİRİR.

- SUM(TOPLAM DEĞERİ GETİRİR.)

  SELECT SUM(AGE) FROM CUSTOMER =1420701 DEĞERİ GETİRİR.

- MAX, MIN (EN BÜYÜK YA DA EN KÜÇÜK DEĞERİ GETİRİR.)

  SELECT MAX(AGE) FROM CUSTOMER = 65
  SELECT MAX(AGE) FROM CUSTOMER = 19 DEĞERLERİNİ GETİRİR.

- COUNT(ÇIKAN SONUCUN SATIR SAYISINI VERİR.)

  SELECT COUNT(AGE) FROM CUSTOMER = 33839

GROUP BY KULLANIMI

Aggregate fonksiyonlarını tablo sorgularımız için kullandığımda group by kullanmam zorunludur.

```
SELECT SUM(LINENET),CLIENTNAME, PHONE FROM CUSTOMER
GROUP BY CLIENTNAME, PHONE
HAVING SUM(LINENET) > 100
```

- HAVING AGGREGATE FONKSİYONLARIN WHERE ŞARTI SAĞLAYICISIDIR. GROUP BY DAN SONRA GELİR.

NOT: Group by kullanımında yazılacak tablo başlıkları Aggregate fonksiyonu dışında kalan tablo başlıkları yazılır.

DISTINCT KOMUTU

Tekrar eden değerleri tekilleştirerek. Benzersiz liste getirir.

```
SELECT DISTINCT(NAMESURNAME) FROM CUSTOMER
```

STRING FONKSİYONLAR

LEFT, RIGHT, UPPER, LOWER, SUBSTRING, LTRIM, RTRIM, REVERSE, REPLACE, CHARINDEX gibi fonksiyonlar metinsel değerler üzerinde işlem yapmamızı sağlar.

SELECT SUBSTRING('SERKAN COŞKUN',1,6) = SERKAN → (VERİLEN ARALIKTAKİ METNİ SEÇER)

SELECT CHARINDEX(' ','SERKAN COŞKUN',1)=7 → (ARANAN DEĞERİN SIRASINI VERİR)

SELECT SUBSTRING('SERKAN COŞKUN',CHARINDEX(' ','SERKAN COŞKUN',1),99) = COŞKUN

SELECT * FROM string_split('YAVUZ, SELİM, KART',',') → METNİ SÜTUNLARA DÖNÜŞTÜRÜR.

SELECT CHOOSE(3,'A','B','C')= C → VERİLEN SIRA NUMARASINA GÖRE DEĞERİ SEÇER.

SELECT STUFF ('SERKAN COŞKUN',8,13,'KAR') = SERKAN KAR. → STUFF DEĞERLERİ DEĞİŞTİR.

SET QUTED_IDENTIFIER ON CREATE TABLE "USER" -> SQL İÇİN AYRILMIŞ ÖZEL KELİMLERİ NESNE OLUŞTURABİLİRİM. " " İÇİNE ALARAK YAZMAMIZ GEREKMEKTEDİR.

SP_RENAME İLE TABLO ADI GÜNCELLEME

Sp_rename ile tablo adında ve kolon adlarında güncelleme yaparım. Exec komutu ile çalışmaktadır.

EXEC SP_RENAME'CUSTOMER','MUSTERILER' CUSTOMER tablo adını MUSTERILER olarak güncellendi.

SAYISAL FONKSİYONLAR

SELECT POWER (DEĞER, ÜST DEĞERİ) SELECT POWER(5,3)=125 → (ÜSSÜNÜ ALIR)

SELECT ABS(MUTLAK DEĞER ALIR)

SELECT SQRT(KAREKÖK ALIR)

SELECT PI() SIN() , COS () , TAN() , COT() (İÇERİ YAZILAN DEĞERLERİN BAŞLIĞA GÖRE DEĞERİNİ GETİRİR.)

SELECT RAND(0-1 ARASINDA DEĞER VERİR) FLOOR (AŞAĞIYA YUVARLAMA İŞLEMİ YAPAR)

SELECT CEILING(7.13) =8 SONUCU GELİR YUKARIYA YUVARLAMA İŞLEMİ YAPAR.

SELECT SIGN(-5) = -1 SIGN İÇERİSİNDEKİ DEĞERİN POZİTİF NEGATİF NÖTR OLDUĞUNU SÖYLER.

SELECT ISNUMERIC(-5) = 1 ISNUMERIC SAYIYSA 1 DEĞİLSE 0

SET STATISTICS TIME ON ÇEKİLEN VERİNİN NE KADAR SÜREDE GELDİĞİNİ SÖYLER: geçen süre = 55484 MS.

SET STATISTICS IO ON SORGUNUN BOYUTUNU KAÇ PAGE VE NE KADAR SÜREDE GELDİĞİ BİLGİSİNİ VERİR.

TARİHSEL FONKSİYONLAR

SELECT GETDATE()→ BUGÜNÜN DEĞERİNİ GETİRİR

SELECT DATEADD(DAY,999,GETDATE()) → SEÇİLİ DEĞİŞKENE(DAY,WEEK,MONTH,YEAR) GÖRE EKLEME YAPAR.

SELECT DATEDIFF(YEAR,('26/05/1998'),GETDATE()) → GÜN AY YIL FARKI ALIR

SELECT DATEPART(MONTH, GETDATE()) → GÜN AY YA DA YILIN KAÇINCI DEĞER OLDUĞUNU HESAPLAR

SELECT DATENAME (WEEKDAY (YA DA MONTH),GETDATE()) → AYIN HAFTANIN GÜNÜN ADINI GETİRME

DML KOMUTLARI

SELECT, UPDATE, DELETE, INSERT (VERİ ÜZERİNDE YAPILAN İŞLEMLER İÇİN KULLANILIR.)

Veri tabanına nesnelerini manipüle etmemizi sağlar

- **SELECT** sorgusu bir insan eli gibidir seçme işlemlerinde kullanılır.

- **INSERT** işlemi tablolara veri kayıt etmek için kullanılır. İlk olarak tablo ve kolon başlıklarımı yazarım daha sonra values fonksiyonu ile ilgili başlıkların sırasına göre değer ataması yaparım.

  ```
  INSERT CUSTOMER (NAMESURNAME, AGE) VALUES ('SERKAN COŞKUN' , 24)
  ```
  CUSTOMER tablosuna bu değerlerimizi kaydetmiş oluruz.

- **UPDATE** Tablomda güncellemeler yapmamı sağlar.
  UPDATE CUSTOMER SET AGE=AGE+1 → Customer tablosundaki tüm AGE değerlerine +1 eklenir
  TOP komutu ile update etmek.
  UPDATE TOP (10) CUSTOMER SET AGE = AGE+1 (İlk 10 değerin yaşına +1 eklenmiştir.)

- DELETE Tablomda silme işlemlerini yapar.

  DELETE FROM CUSTOMER (Tüm customer tablosunu siler.)
  WHERE ID = 78 -> 78. ID Numarasını siler. (Where ile şarta bağladığım sonuçları siler.)

- INTO **ile** tablo yedeği alma

  SELECT * INTO CUSTOMERYEDEK FROM CUSTOMER komutu ile
  CUSTOMER tablosundaki (* yazdığımız için) tüm başlıklarını CUSTOMERYEDEK tablosuna atmaktadır.

- TRUNCATE Tablolarımın içini boşaltmamı sağlar. Identity kolonunu sıfırlar.

  ```
  TRUNCATE TABLE CUSTOMER
  ```

DDL KOMUTLARI

CREATE, ALTER, DROP (TABLO ÜZERİNDE YAPILAN İŞLEMLER İÇİN KULLANILIR.)

- DDL komutları veri tabanına nesneleri oluşturmamızı sağlar. Bu nesne üzerinde değişiklik ve silme işlemleri yapmamızı sağlamaktadır.

- **CREATE** Database, tables, view, storedproc, trigger vs. oluşturmamızı sağlar.
  ```
  CREATE TABLE CUSTOMER ( ID INT , NAMESURNAME NVARCHAR(100) , AGE INT )ile CUSTOMER tablosu ve ID,NAMESURNAME,AGE kolonları eklenmiştir.
  ```
- ALTER Create ile yapılan veri tabanı nesneleri günceller. Mevcut tablodaki sütunlara ekleme silme değiştirmede kullanılır.

  Alter ile constraint eklenebilir.

  ALTER TABLE CUSTOMER
  ADD SET DROP KOLON4 INT (ADD İLE EKLEME, SET DÜZENLEME VE DROP İLE KALDIRMA İŞLEMİ YAPILIR.)

- **DROP** Create ile oluşturulanı silmemizi sağlamaktadır. Tabloyu kaldırmadan sadece satırlarını boşaltmak istiyorsanız DELETE kullanın. DROP TABLE daima hedef tablonun içerdiği indeksleri, kuralları, tetikleri ve kısıtları kaldıracaktır.
  DROP TABLE CUSTOMER

WITH KOMUTU

WITH NOLOCK

SQL serverda kullanıcı tablo'da bir kaydı açıp üzerinde işlem yapıyorsa o kayıt SQL server tarafından kilitlenmektedir. Aynı kayıt üzerinde işlem yapmak isteyen kullanıcılar diğer kullanıcının işleminin bitmesini beklemek durumundadır. With nolock komutu tabloların kilitlenmesini engelleyen bir yapıdır. Bu sayede aynı tablo üzerinde aynı anda birden fazla kullanıcı işlem yapabilmektedir.
Büyük ölçekli projelerde kullanılması tavsiye edilir.

```
SELECT * FROM CUSTOMER WITH (NOLOCK)
```

WITH ROLLUP

Group by ile kümelenmiş veri kümesinde **ara toplamı** verir. Group by dan sonra gelir.

```
SELECT CLIENTNAME, CATEGORY_NAME1,SUM(LINENET) FROM SALES

GROUP BY CLIENTNAME, CATEGORY_NAME1 WITH ROLLUP
```

Kullanıcının yaptığı alışverişlerin kategori bazlı toplam tutarlarını verir.

WITH CUBE

Group by ile kümelenmiş veri kümesini teker teker toplar.

```
SELECT CLIENTNAME, CATEGORY_NAME1,SUM(LINENET) FROM SALES

GROUP BY CLIENTNAME, CATEGORY_NAME1 WITH CUBE
```

Kategorilerin kullanıcılara göre alışveriş toplamını verir.

WITH TIES

Top N ile şart sağladığım satırlara sıralama yapar ancak bu sıralama eşit olan değerleri aynı sıraya ekler.

```
SELECT TOP 3 WITH TIES LINENET FROM SALES ORDER BY LINENET
```

Order by ile sıralanmasını istediğim başlığı eklerim.

WITH İLE TABLO OLUŞTURMA

Karmaşık sorguları tek çatı altında toplamayı sağlar. As den sonra gelen sorgunun adı CTE dir.

```
WITH CTE
AS(
  SELECT ID, ITEMNAME
    , RN=ROW_NUMBER()OVER(ORDER BY ID)
  FROM CUSTOMER
)
UPDATE CTE SET ID=RN
```

ID Kolonunu yeni baştan dizmek istersem ve ada göre sıralanmasını istersem With komutunda RN takma isim verdiğim için update işleminde kullanmamı sağlamaktadır.

Bu tarz karmaşık sorgularda With ile oluşturacağımız tablolara vereceğimiz aliasları DDL ve DML işlemlerde kullanabilmekteyiz.

SIRALAMA VE DÜZENLEME KOMUTLARI

ORDER BY

Sıralama işlemi yapmaktadır. Order by dan sonra sıralanacak kolon adı girilir ve eğer küçükten büyüğe sıralamak istersek asc, Büyükten küçüğe sıralamak için desc kullanılır. Ancak kolon adından sonra bir şey yazmayıp kodu girersek otomatik küçükten büyüğe sıralayacaktır.

```
SELECT * FROM CUSTOMER ORDER BY AGE DESC
```

ROW_NUMBER FONKSİYONU

Order by dan sonra verilen başlığa göre id değeri sıralar.

```
SELECT ROW_NUMBER() OVER(ORDER BY AGE) AS NUMBERAGE , * FROM CUSTOMER
```

PARTITION BY KOMUTU

Örnekte önce ada göre sıralayıp sonra yaşa göre sıralamış ve id değerlerini verir.

```
SELECT ROW_NUMBER () OVER(PARTITION BY NAMESURNAME ORDER BY AGE) AS INDEXAGE , *
FROM CUSTOMER
```

ROW NUMBER - RANK- DENSE_RANK

Özel sıralama işlemi yapar.
Rank 11144448888 rank adil sıralama yapar.
Dense_rank 11112223333 ise eşit sıralama yapmaktadır.

```
SELECT ID, COUNT(*) AS "Satış Sayısı",
RANK() OVER(ORDER BY COUNT(*) DESC) AS "Rank ile Sıralama",
ROW_NUMBER() OVER(ORDER BY COUNT(*) DESC) AS "Normal Sıralama",
DENSE_RANK() OVER(ORDER BY COUNT(*) DESC) AS "Dense_Rank ile Sıralama"
FROM CUSTOMER
GROUP BY ID
ORDER BY COUNT(*) DESC
```

https://www.ismailgursoy.com.tr/rank-ve-dense_rank-fonksiyonlari/

OFFSET –ROWS –FETCH NEXT-ROWS ONLY İLE ARA DEĞERLERİ GETİRME

Yazılan sorguda offsetin yanına gelecek rakam kaçtan başlayacağı next in yanına gelen rakam ise kaç adet alacağım anlamına gelir.

```
SELECT ID, NAMESURNAME
FROM CUSTOMER
ORDER BY ID
OFFSET 5 ROWS FETCH NEXT 20 ROWS ONLY;
```

ID değeri 5 den sonra olan 20 değeri getirmektedir.

NTILE

Parantez içinde rakama göre bölüm yapar ve sıra verir.

```
SELECT ID, NTILE(2) OVER(ORDER BY (ID)) FROM CUSTOMER
```

Customer tablosunu 2 eşit parçaya bölüp 1 ve 2 değerlerini atamaktadır.

IDENTITY İLE İLGİLİ KOMUTLAR

IDENTITY OLUŞTURMA

```
CREATE TABLE SERKAN (
ID INT IDENTITY(1,1) )  1 ER 1 ER ARTSIN 1 DEN BAŞLASIN ANLAMLARINA GELİR
```

@@IDENTITY KOMUTU

İlgili veri tabanında **İNSERT** edilen değerin identity değerini verir.

INSERT CUSTOMER DEFAULT VALUES -> DEFAULT VALUES DEĞER YOK ANLAMINA GELİR

NAMESURNAME BIT DEFAULT '1' -> İÇİNE HANGİ DEĞERİ GİRERSEM GİREYİM 1 DEĞERİ GELİR

SELECT @@IDENTITY = 18780 değeri gelir.

Trigger kullanılan sorgularda yanlış sonuç alma ihtimalinden dolayı kullanılması tavsiye edilmez.

SCOPE_IDENTITY KOMUTU

SELECT SCOPE_IDENTITY() -> Son id değerini getirir.

IDENT_CURRENT

SELECT IDENT_CURRENT('CUSTOMER') İnsert işleminden sonra devam edecek id değerini sırasını verir.

DBCC CHECKIDENT İLE IDENTITY KOLONUNA MÜDAHALE ETME

Identity değerimin kaçıncı değerden başlamasını veya devam edeceğini sağlar.

DBCC CHECKIDENT (CUSTOMER, RESEED,78) -> Reseed sorgunun içinde bulunan parametredir.
"78 " ise devam etmesini istediğim id değerimdir.

@@ROWCOUNT

Yapılan işlemde etkilenen satır sayısını getirir.

```
SELECT * FROM CUSTOMER
SELECT @@ROWCOUNT =18768 değeri gelir. Bu sorgumda 18768 değer etkilenmiştir.
```

NULL DEĞER KONTROLÜ

ISNULL

Null değeri gelen sonuçları parantezin virgülden sonraki kısmına verilen değeri yazar.

```
SELECT ISNULL(CITY,'ŞEHİR BİLGİSİ YOK') FROM CUSTOMER
```

NULL değeri yerine ŞEHİR BİLGİSİ YOK değeri gelmektedir.

COALESCE

Mantık olarak isnull ile aynıdır. Null değeri yerine atayacağımız değeri getirir.

```
SELECT COALESCE(CITY,'ŞEHİR BİLGİSİ YOK') FROM CUSTOMER
```

DEĞİŞKEN TANIMLAMA

Değişkenleri Declare ile tanımlayabilirim.

```
DECLARE @x INT = 5 , @Y NVARCHAR(MAX) = 'AKİF'
```

Declare ile atadığımız değeri select ile çağırabiliriz.

```
SELECT @X,@Y
```

	(No column name)	(No column name)
1	5	AKİF

Sonucu gelmektedir.

Tanımlanmış değişkene set ile de değer atayabiliriz.

```
DECLARE @x INT = 5 , @Y NVARCHAR(MAX) = 'AKİF'
SET @X=@X+7
SELECT @X,@Y
```

ÖR: Yaşı @x olarak tanımladığımız değerleri getirme.

```
DECLARE @x INT=25  , @Y NVARCHAR(MAX) = 'AKİF'
SELECT AGE FROM CUSTOMER WHERE AGE=@X
```

DECLARE İLE TABLO OLUŞTURMA

```
DECLARE @Musterilerimiz TABLE
(MusteriID INT, Ad VARCHAR(50), Soyad VARCHAR(25))
```

DECLARE İLE OUTPUT PARAMAETRESİ

Insert veya delete yapılan işlemleri etkilenen satırları getirir.

```
DECLARE @SİLİNEN TABLE (CITY VARCHAR(MAX) )  -- CITY SİLECEĞİMİZ TABLODA ÇEKMEK İSTEDİĞİMİZ SÜTUN ADI
DELETE FROM CUSTOMERYEDEK
OUTPUT DELETED.CITY
INTO @SİLİNEN -- INTO İLE SİLİNEN DEĞERLERİ @SİLİNEN DEĞİŞKENİNE AKTARIRIZ.
WHERE CUSTOMERYEDEK.AGE > 40 -- İSTENİLİRSE WHERE ŞARTIDA VERİLEBİLİR.
SELECT * FROM @SİLİNEN
```

AKIŞ KONTROLLERİ

IF FONKSİYONU YAPISI

MS SQL'de T-SQL yapısında da koşullu sorgular oluşturma ihtiyacı duyabiliriz. Tıpkı programlama dillerinde olduğu gibi koşul sonucu TRUE ve FALSE mantıksal yapısına göre işlemler ele alınırlar.

=,<>,>,< şartlarını sağlayıp olumlu ve ya olumsuz durumlarda etki etmesini sağlar

IF ELSE NİN TABLO ÜZERİNDE KULLANIŞI

```
DECLARE @X INT = 1000
DECLARE @Y INT
SELECT @Y=AGE FROM CUSTOMER WHERE ID=@X
IF @Y>30
BEGIN
PRINT 'YASLIDIR'
END
ELSE
BEGIN
PRINT 'YASLI DEĞİLDİR'
END
SELECT @Y
```

MESSAGES KISMINDA YASLIDIR DEĞERİ GELMEKTEDİR.

ÖR: Değişkene direkt olarak değer atayıp where şartında bu değeri döndürürsem birden fazla değer gelmektedir.

```
DECLARE @X NVARCHAR(MAX) = 'KARABÜK'
IF @X IS NOT NULL
SELECT * FROM CUSTOMER WHERE CITY=@X
```

EXISTS FONKSİYONU YAPISI

Aratılan sorguda değer geliyorsa TRUE gelmiyorsa FALSE olarak bilgi döner.

```
IF EXISTS (SELECT * FROM CUSTOMER)
PRINT  ' DOLU'
ELSE
PRINT 'BOS'
```

https://bitimek.com/exists-ve-not-exists-kullanimi/

IIF FONKSİYONU

Mantık olarak Exceldeki eğer mantığıyla aynıdır. Eğer yapısı, doğruysa, yanlışsa olarak dizilmektedir.

```
DECLARE  @X INT = 10
SELECT IIF(@X>5,'5 TEN BÜYÜK','5 TEN KÜÇÜK')
```

→ 5 TEN BÜYÜKTÜR DEĞERİ GELİR.

CASE WHEN ELSE END FONKSİYONU

Mevcut bir veri satırında yer alan değerin daha sonra istediğimiz şekilde görüntülenmesini sağlayabiliriz.

CASE WHEN ELSEDE UPDATE NASIL KULLANILIR

```
UPDATE CUSTOMER SET YASLI=
CASE
WHEN AGE<20 THEN 'ÇOCUK'
WHEN AGE<30 THEN 'GENC'
WHEN AGE<40 THEN 'OLGUN'
WHEN AGE>40 THEN 'YASLI'
ELSE ' '
END
WHERE AGE BETWEEN 0 AND 100
```

```
SELECT
CASE ABS (AGE) % 4
WHEN 0 THEN 'Dr.'
WHEN 1 THEN 'Master'
WHEN 2 THEN 'Mr'
WHEN 3 THEN 'Mrs'
END
FROM CUSTOMER
```

Sayısal olarak Aggregate fonksiyonlarda kullanabilir

Yaşı 40 dan büyükler yaşlı 40–30 arası olgun, 30-20 arası genç ve 20 den küçükleri çocuk olarak güncellenmiştir.

CASE WHEN ELSE ORDER BY İLE KULLANIMI VE ÖRNEKLERİ

Case ile order by kullanarak sütun gizleyerek sıralama yapan bir sorgu

```
SELECT NAMESURNAME FROM CUSTOMERYEDEK
ORDER BY CASE
WHEN CITY = 'İSTANBUL' THEN 1
WHEN CITY ='ANKARA' THEN 2
WHEN CITY ='BURSA' THEN 3
WHEN CITY ='ÇANAKKALE' THEN 4
WHEN CITY ='KARABÜK' THEN 5
ELSE 99
END
```

CITY değerlerine sıralama olarak atabiliriz. Bu sayede City değeri İstanbul olanlar 1. Sırada Ankara olanlar 2. Sırada ve diğerleri verdiğimiz sıraya göre dizilecektir.

EĞER İSTERSEM CASE İN YANINA DİREK CITY YAZIP CITY I TANIMLAYABİLİRİM.

```
SELECT CASE CITY
WHEN 'İSTANBUL' THEN 1
ELSE 99
END CITY, NAMESURNAME
FROM CUSTOMERYEDEK
```

WHILE DÖNGÜSÜ İLE ŞART KOMUTU

-İken sorusunun şartlanmasını sağlar. Döngü parametresidir.

```
DECLARE @SAYAC INT = 0 → SAYAC'A 0 DEĞERİNİ ATARIZ.
WHILE @SAYAC < 100          (@ SAYAC 100 DEN KÜÇÜK-İKEN)
BEGIN
SET @SAYAC=@SAYAC+1         SONUC OLARAK 1 DEN 100 E KADAR SIRALI DEĞERLER GELİR
PRINT @SAYAC
END
```

WHILE DÖNGÜSÜNDE BREAK KOMUTU

Break şartı sağlandığında işlemi durdurur

```
DECLARE @SAYAC INT = 0
WHILE @SAYAC < 100
BEGIN
SET @SAYAC=@SAYAC+1
IF @SAYAC %5 = 0              5 MODU 0 OLDUĞUNDA BIRAK
BREAK
PRINT @SAYAC
END                 Sonuç olarak 1,2,3,4 gelir 5. Değerde işlemi bırakır.
```

WHILE DÖNGÜSÜNDE CONTINUE KOMUTU

Continue komutu şartı **sağlamayanı eklemez** ve devam eder.

```
DECLARE @SAYAC INT = 0
 WHILE @SAYAC < 100
  BEGIN
   IF @SAYAC %5 = 0              MODU 5 OLAN DEĞERLERİ GETİRMEYİP DEVAM EDECEK.
    BEGIN
          SET @SAYAC=@SAYAC+1
CONTINUE
    END
 PRINT @SAYAC
 SET @SAYAC=@SAYAC+1
END
SONUÇ OLARAK 1,2,3,4,6,7,8,9,11,...,99 OLARAK GELİR

--DÖNGÜSEL VERİ OLUŞTURMA

 CREATE TABLE TABLO1
 (
 ID INT IDENTITY(1,1),
 AD VARCHAR(20))
GO
DECLARE @SAYAC INT =1
WHILE @SAYAC < 250001
BEGIN
INSERT TABLO1
SELECT 'KİTAP' + CAST(@SAYAC AS VARCHAR(20))
SET @SAYAC=@SAYAC+1
END
SELECT * FROM TABLO1
SELECT COUNT(*) FROM TABLO1
```

PIVOT

- Pivot genellikle raporlama alanlarında kullanılmaktadır.
- Alt alta olan sıralamaları yan yana göstermemizi sağlamaktadır.
- Ör. Şirketimizin Türkiye'nin farklı bölgelerde mağazaları var ve çalışanlarımızın hangi bölgelerde ne kadar satış yaptığını görmek istediğimiz bir tablo oluşturmamızı istiyor.

	REGION	SALESMAN	TOPLAMSATIS
1	Akdeniz	Metehan EĞİNKAYA	3359,27
2	Karadeniz	Ramazan SULAK	1270,05
3	Karadeniz	Emre PLAKET	1477,67
4	İç Anadolu	Gülüzar KURUDAN	822,84
5	Güneydoğu Anadolu	Esila BEZİRGANOĞLU	1617,83
6	Akdeniz	Meral DUDULAROĞLU	5370,34
7	Doğu Anadolu	Sevim PEHLİVANOĞLU	1642,44
8	Doğu Anadolu	Şeyma HOBAPLI	3734,24
9	Akdeniz	Erdal PÜSKÜLLÜ	8223,53
10	Doğu Anadolu	Yaren KARAKAYALI	1885,05
11	İç Anadolu	Buse ÇINAR	1086,44
12	İç Anadolu	Mehmet Ali PADAR	1550,85
13	Karadeniz	Defne ŞAHAN	1356,59
14	Akdeniz	Birgül SAYGIN	1107,92

- Raporumuzun anlamlılığı açısından BRANCH ları birer sütun olarak almak istiyorsak bu kısımda PİVOT komutu devreye girmektedir. Pivot komutu sayesinde branch ları sütun olarak getirebiliriz.

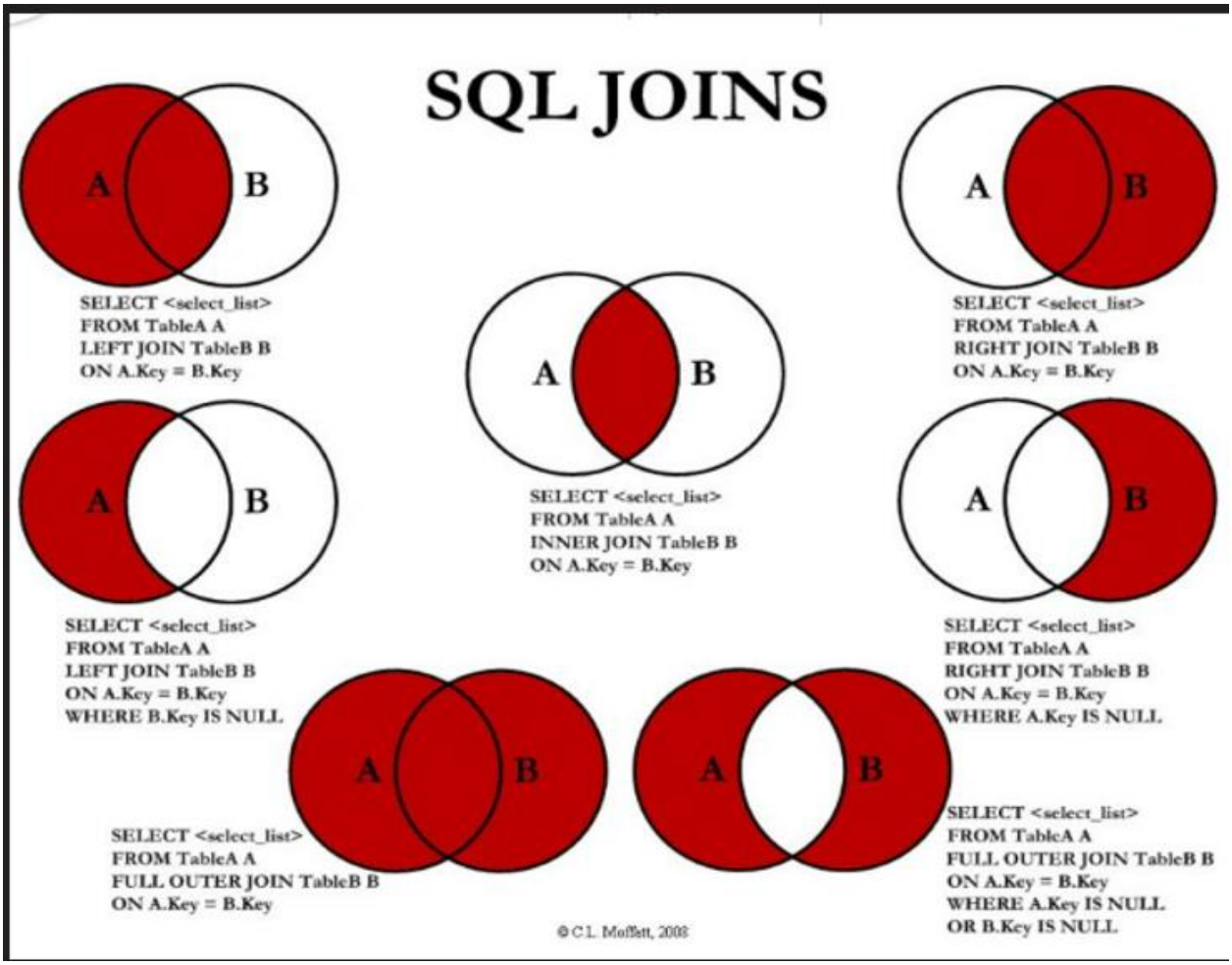
	SALESMAN	NULL	Akdeniz	Doğu Anadolu	Ege	Güneydoğu Anadolu	İç Anadolu	Karadeniz	Marmara
1	Aleyna ÇUKUREL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2651,88
2	Aliye KÖZEN	NULL	NULL	326,95	NULL	NULL	NULL	NULL	NULL
3	Ayaz SABANCIOĞLU	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1782,56
4	Bedriye KÜÇÜKÖZER	NULL	13205,16	NULL	NULL	NULL	NULL	NULL	NULL
5	Belinay BİLGEN	NULL	NULL	NULL	NULL	NULL	19410,26	NULL	NULL
6	Berkay SAN.TİC	NULL	NULL	NULL	4570,3	NULL	NULL	NULL	NULL
7	Berna SARIÇİÇEK	NULL	NULL	NULL	6592,22	NULL	NULL	NULL	NULL
8	Berra DÜNDAR	NULL	NULL	NULL	NULL	2111,44	NULL	NULL	NULL
9	Buse ÇINAR	NULL	NULL	NULL	NULL	NULL	1086,44	NULL	NULL
10	Bünyamin ÇAĞIR	NULL	NULL	NULL	NULL	NULL	NULL	768,49	NULL
11	Dilan ÇALAR	NULL	NULL	NULL	NULL	NULL	NULL	NULL	69508,8000000001
12	Dilan LORAS	NULL	NULL	NULL	NULL	NULL	709,28	NULL	NULL
13	Doğukan BAHÇIVANCI	NULL	NULL	NULL	NULL	NULL	NULL	572,34	NULL
14	Ebrar OGUZPERDAHÇI	NULL	NULL	NULL	NULL	1917,83	NULL	NULL	NULL
15	Ecrin ALAKAŞ	NULL	NULL	NULL	NULL	NULL	NULL	2590,5	NULL
16	Eda SINAV	NULL	NULL	NULL	NULL	NULL	18827,16	NULL	NULL

- **Önemli:** Pivot fonksiyonu büyük boyutlu verilerde performans açısından oldukça maliyetli olabilir. Eğer ki performans söz konusu olan ama bir yandan da milyonlarca veri üzerinde işlem yapmanızı gerektiren raporsal çalışmalarınızda bu maliyeti göz önünde tutarak hareket etmenizi tavsiye ederim.

JOINLER

SQL de birbiri ile ilişkili tablolardan veri çekmek için JOIN komutlarını kullanarak ilişkili tabloları tek çatı altında toplarız. Joinler bire bir, Bire çok, Çoka çok olarak ilişkilenmektedir.

JOINLERİN GRAFİKSEL GÖSTERİMLERİ



https://www.ismailgursoy.com.tr/sqlde-bilinmesi-gereken-en-temel-konular/

4 tipte join vardır.

1. INNER JOIN: Her iki tablodaki eşleşen kayıtlar yalnızca geriye döner. Bunu şöyle düşünün a ve b kümesi olsun iki kümenin birleşimi inner joindir.

2. LEFT JOIN: İki tablo içerisinde solda yer alan tablonun kayıtları ve sağdaki tabloda eşleşen kayıtlar döner.

3. RIGHT JOIN: İki tablo içerisinde sağda yer alan tablonun kayıtları ve soldaki tabloda eşleşen kayıtlar döner.

4. FULL JOIN: İki tablo birleştirilir ve sonucunda tüm kayıtlar döner.

LEFT OUTER JOIN

LEFT OUTER JOIN HAKKINDA BİLGİ

https://www.ismailgursoy.com.tr/left-outer-join-fonksiyonunun-ikiden-fazla-tablo-icin-kullanimi/

OUTER JOIN VE RIGHT JOIN

OUTER JOIN İLE İNNER ARASINDAKİ FARKI ANLATILIYOR.

https://www.ismailgursoy.com.tr/left-outer-join-ve-right-outer-join/

JOINLERİN İÇİNDE SUM KULLANIMI

JOIN TABLOSU İÇİNDE SORGUYA SUM EKLEME

https://www.ismailgursoy.com.tr/turetilmis-tablo-kullanarak-toplama-fonksiyonlarinin-sorgu-icinde-ayrilmasi/

DEĞİŞKENLE TABLO OLUŞTURMA

DEĞİŞKENİ TABLO OLUŞTURUP JOINLE DİĞER TABLOYU İÇE AKTARIYOR

https://www.ismailgursoy.com.tr/tablolarin-degisken-olarak-olusturulmasi/

SUBQUERY (İÇ İÇE OLAN SORGULAR)

Karmaşık sorgular için kullanacağımız sorgu içinde sorgulardır.

Subquery i kullanmamızın en doğru yöntemi önce parantez içinde oluşturduğumuz sorgunun çalışıp hangi değerleri getirdiğine bakarız daha sonra bu değer ile etkilenen sorgu ya da sorguların sonuçlarına bakarız.

Subquery i sorgularında önemli olan en alt sorgudan üst sorguya giderek kontrol sağlamamızdır.

SUBQUERY ÖRNKERLERİ

https://www.ismailgursoy.com.tr/subquery-alt-sorgu-ornekleri/

SUBQUREY İ İLE UPDATE KULLANIMI

```
UPDATE
Employees
SET PhoneNumber =
(SELECT c. PhoneNumber
FROM
Customers c
WHERE
C. FName Employees. FName
AND c. LName = Employees. LName) WHERE Employees. PhoneNumber IS NULL
```

SUBQUERY İÇİNDE JOIN KULLANIMI

```
SELECT po. Id, po. PODate, po. Vendor Name, po. Status, item. ItemNo,
item. Description, item. Cost, item. Price
FROM PurchaseOrders po
LEFT JOIN
SELECT 1.PurchaseOrderId, 1.ItemNo, 1.Description, 1. Cost, 1 Price, Min(1 id)
FROM PurchaseOrderLineItems 1 GROUP BY 1. PurchaseOrderId, 1.ItemNo, 1.Description, 1. Cost, 1. Price)
AS item ON item. PurchaseOrderId = po. Id
```

SUBQUERY VE JOINLER

```
SELECT NAMESURNAME, SAYI FROM CUSTOMER WHERE ID IN
( SELECT ID FROM CUSTOMERYEDEK WHERE SAYI IS NULL)
```

https://www.ismailgursoy.com.tr/birden-fazla-tablodan-veri-sorgulamak/

SUBQUERY İÇİNDE SUM KULLANIMA

https://www.ismailgursoy.com.tr/toplama-fonksiyonlarinin-alt-sorgular-icinde-select-ile-birlikte-kullanilmasi/

SUBQUERY IN VE NOT IN KULLANIMI

https://www.ismailgursoy.com.tr/in-ve-not-in-listelerinde-alt-sorgu-kullanimi/

CROSS APPLY (KESİŞİM KÜMESİNİN) KULLANIMI

Subqery içine yazılan sorgunun üstüne yazılır ve left ya da right outer joinli sonucun aynı değerlerini getirir.

```
SELECT U.NAMESURNAME FROM USER_ U
CROSS APPLY
( SELECT STATUS_ FROM BASKET B WHERE B.USERID=U.ID) ÖRNEK
```

Birden fazla sonuç gelecek sorgularımda where şartının önüne IN yazarım.

UNION VE UNION ALL KOMUTU

İki veya daha fazla tabloyu select sorgusunda (sütun sayıları eşit olma şartı ile) tablolarımı alt alta bağlar.

```
SELECT * FROM CUSTOMER
UNION - UNION ALL
SELECT * FROM CUSTOMERYEDEK
```

"UNION İLE UNION ALL ARASINDAKİ FARK UNION ALL İKİ TABLOLDA DA OLAN DEĞERİ TEK BİR SEFER GETİRİR. UNION ALL TEKRAR EDEN DEĞERLERİ GETİRMEZ."

INTERSECT VE EXCEPT KULLANIMI

INTERSECT iki tablodaki ortak kesişimi verir.

Except iki tablodaki kesişim olmayan kısmı verir.

```
SELECT * FROM CUSTOMER
INTERSECT/EXCEPT
SELECT * FROM CUSTOMERYEDEK
```

Ya da bu komutlar yerine where şartı içerisinde kesişimli alanımız hangi sütunsa Subquery ile aşağıdaki örnek gibi çekebiliriz.

```
SELECT * FROM CUSTOMER WHERE ID IN ( SELECT ID FROM CUSTOMERYEDEK )
```

Eğer kesişmeyen alanı istersem `NOT IN` komutu yazarım

```
SELECT * FROM CUSTOMER WHERE ID NOT IN ( SELECT ID FROM CUSTOMERYEDEK )
```

MERGE KULLANIMI

SQL Serverda merge komutu iki tablo arasındaki farklı gördüğü alanı insert edip aynı gördüğü alanı güncelleyip, farklı alanı ise silebileceğimiz bir yapı oluşturur. İki tablo arası eşitleme gibide düşünebiliriz. Eğer kayıt var ise güncelle, yok ise ekle mantığıyla çalışmaktadır.

```
MERGE INTO CUSTOMERYEDEK
AS YEDEKLEME
USING(VALUES('AYDIN COŞKUN','MARAŞ',27,'GENC',NEWID(),1)) AS AD
(NAMESURNAME, CITY, AGE, YASLI, OZEL, SAYI)
ON YEDEKLEME. AGE=AD. AGE
WHEN MATCHED THEN UPDATE SET YEDEKLEME. NAMESURNAME=AD. NAMESURNAME, YEDEKLEME.
CITY=AD. CITY,
YEDEKLEME. YASLI=AD. YASLI, YEDEKLEME. OZEL=AD. OZEL, YEDEKLEME. SAYI=AD. SAYI
WHEN NOT MATCHED THEN INSERT (NAMESURNAME, CITY, AGE, YASLI, OZEL, SAYI)
VALUES (AD. NAMESURNAME, AD. CITY, AD. AGE, AD. YASLI, AD. OZEL, AD. SAYI);
```

Her iki tabloya da kaydı ekledi eğer birinde bu kayıt olsaydı diğerine ekleyecekti sadece.

https://muhammetaliugurlu.wordpress.com/2013/07/18/sql-server-da-merge-komutu-kullanimi/

GEÇİCİ TABLOLAR # VE

Birden fazla kişini çalıştığı durumlarda ya da verilerin test amaçlı geçici bir yerlerde tutulması işlenmesi amacıyla kullanılan yapılardır.
Geçici tablolar Temp db'de toplanmaktadır.
ile belleğe geçici olarak kaydedilir ve sql ya da PC kapatıldığında tablom silinir.

```
SELECT * INTO # FROM CUSTOMER
```

Tablolarımda gözükmez ancak tüm ddl ve dml işlemlerini bu tabloya yapabilirim.

ile geçici tablo oluşturmanın farkı bu tabloya 3. şahıs bağlanabilir ve 3. Şahıs oturumu kapattığında tablom silinir.

```
SELECT * INTO ## FROM CUSTOMER
```

UNIQUEIDENTIFIER

Benzersiz bir değer oluşturur. Tablolarımda Uniq bir kolon ekleyeceksek bu değer ile o kolona değer ekleyebilirim.

```
SELECT NEWID()
INSERT CUSTOMER(OZEL) VALUES (NEWID())
```

GRANT DENY REVOKE KOMUTLARI

Grant: Kullanıcılara veri tabanı nesneleri üzerinde güvenlik ayrıcalıkları vermek için kullanılan komuttur.

```
GRANT INSERT, UPDATE, DELETE TO BAYRAKTAR
GRANT SELECT ON Bilimkurgu TO BAYRAKTA
GRANT CREATE TABLE TO BAYRAKTAR
GRANT (all | izinler) ON (izneTabiTutulanlar) TO (izinVerilenler)
```

DENY

GRANT komutunun tersidir. Yetkileri engeller. Yukarıdaki örnekle aynı şekilde kullanılabilir.

REVOKE

GRANT ile değiştirdiğimiz hakları eski haline döndürmek için kullanılır. Bir nesneyi oluşturan kullanıcının REVOKE ile nesne üzerindeki yetkilendirme ve kullanma hakkı yok edilemez.

```
REVOKE ALL ON REGION TO BAYRAKTAR
```

DYNAMIC DATA MASKING (SAKLAMA GÜVENLİK)

Select sorgusuyla gelen verilerin fiziksel halini maskeleme işlemi yapmamızı sağlar.

TC no mail adresi telefon numarası adres gibi özel bilgileri maskeleriz.

```
CREATE TABLE CUSTOMERA
( NAMESURNAME NVARCHAR(10) MASKED WITH (FUNCTION='DEFAULT()') NULL,
  EMAIL NVARCHAR(MAX) MASKED WITH (FUNCTION='EMAIL()') NULL,
  TELNO NVARCHAR(12) MASKED WITH (FUNCTION='PARTIAL(4,"******",1)') NULL)
  --  PARTIAL İLK 4 KARAKTERDEN SONRA * İLE 1 ADET MASKELEME YAPACAK


  CREATE USER YETKILIUSER WITHOUT LOGIN -- YETKİLİ İSMİNDE KULLANICI OLUŞTURUYORUM.
  GO
  GRANT SELECT ON CUSTOMERA TO YETKILIUSER -- YETKILIUSER İSİMLİ KULLANICIYA SELECT YETKİSİ
VERDİM
  CUSTOMERA İÇİN

  EXECUTE AS USER ='YETKILIUSER' -- YETKILIUSER İSİMLİ KULLANICIDAYKEN
  SELECT * FROM CUSTOMERA
```

ALTER İLE MASKELEME EKLEME FORMATI DEĞİŞTİRME:

```
ALTER TABLE CUSTOMERA
ADD EKKOLON NVARCHAR(MAX) MASKED WITH (FUNCTION='DEFAULT()')NULL
```

ALTER İLE MASKELEME KALDIRMA

```
ALTER TABLE CUSTOMERA
ALTER COLUMN EKKOLON DROP MASKED
```

KULLANICIYA GÖRE D.D.M. ÖZELLİĞİNİ PASİFLEŞTİRME

```
GRANT UNMASK TO YETKILIUSER
```

CONSTAINTLER(KISITLAYICILAR)

DEFAULT CONSTRAINT

Default constrainte eklediğimiz kolona değer girilmezse kısıtlayıcı sayesinde Null değer yerine istediğimiz değeri girecektir.

```
ALTER TABLE CUSTOMER
ADD CONSTRAINT CS_BOŞ DEFAULT'BOŞ' FOR NAMESURNAME
```

Artık NAMESURNAME kolonuna değer girilmediğinde BOŞ olarak değer atamaktadır.

CHECK CONSTRAINT

Check constraint: Girilecek verinin şartlara uygun olmasını ister aksi halde kayıt yaptırmaz.

```
ALTER TABLE CUSTOMER
WITH NOCHECK
ADD CONSTRAINT CS_KNTRL CHECK (AGE>18) BU SAYEDE AGE KOLONUNA 18 YAŞ VE ALTI GİRİLEMEYECEK
```

- CONSTRAINT AYARLAMALARI YAPARKEN ÇAKIŞMA HATASI ALIYORSAK BUNUN SEBEBİ CONS ÖNCESİNDEKİ VERİLERİN ŞARTLARI SAĞLANMAMASIDIR. BU HATAYI AŞMAK İÇİN WİTH NOCHECK KOMUTUYLA AŞABİLİRİZ.

PRIMARY KEY CONSTRAINT

Tabloda primary key olmamalıdır ve sadece 1 adet primary key olmalıdır. Bu işlemi sorgu ile yapabilirim.

```
ALTER TABLE CUSTOMER
ADD CONSTRAINT PRIMARYID PRIMARY KEY (ID)
```

UNIQUE CONSTRAINT

Tek amacı belirtilen kolondaki değerleri tekil olmasını sağlar. Tekrar edecek değerleri insert etmez.

```
ALTER TABLE CUSTOMER
ADD CONSTRAINT UNIQUZEL UNIQUE (OZEL)
```

FOREIGN KEY CONSTRAINT

İlişkisel tablolarda kullanılır ilişkisel değerleri silme işlemleri engeller.

```
ALTER TABLE OGRENCILER
ADD CONSTRAINT FOREIGN_OGRENCILER FOREIGN KEY (DERSID) REFERENCES DERSLER(DERSID)
```

- ON DELETE CASCADE YA DA ON UPDATE CASCADE(İKİSİDE BİR ÇALIŞABİLİR) İLE FOREİGN KEY CONSLARDA İLİŞKİLİ SATIRLARI SİLMEMİZİ SAĞLAR

- ON DELETE CASCADE KULLANIMLARINDA DEFAULT CONS U İLE ÇAKIŞIP ÇAKIŞMADIĞINA DİKKAT ETMELİYİZ.

```
ALTER TABLE OGRENCILER
ADD CONSTRAINT FOREIGN_OGRENCILER FOREIGN KEY (DERSID) REFERENCES DERSLER(DERSID)
ON DELETE CASCADE -> SİLME İŞLEMİNDE HER İKİ TARAFIDA SİLER.
ON UPDATE SET DEFAULT -> DEFAULT CONSTA KARŞILIK GELENİ EKLER.
```

DİYAGRAMLAR

- Diyagramlar ilişkisel tabloları bağlamamızı sağlar. Diyagramlar aslında Foreign key constraintlerdir.

- İlgili database i seçtikten sonra alt başlığında Database Diagramsları mevcuttur. New diyerek yeni diyagram ekleyebilirim. Diyagramları birbirlerine bağlamam için bağlayacak tablonun **primary key** olması gerekmektedir.

- Bağlamak için ilgili tablonun başlığını seçerek bağlayacağım gösterir.

- Tablonun ilişkili başlığına sürüklerim. Anahtar işareti bir , ∞ işareti ise çok anlamına gelmektedir. Bire çok bir ilişki olduğunu

- Artık tablolar arasında bir ilişki mevcuttur bu sayede ilişkili tabloda silme işlemleri yapamayız. Bu ilişkili tablolarda join ile tablolar arasında bağlama yapabiliriz.

VIEW

- Karmaşık sorguları tek bir sorguya indirmeyi sağlar. Raporlama ve güvenlikte gerekir. Komplikeyi sadeleştirir.
- Herhangi bir sorgunun sonucu tablo olarak ele alır ve bu oluşturulan tablodan sorgu çekebiliriz.
- Fiziksel bir yapıdır. Veri tabanına kaydolur.
- View da içeri insert update delete gibi işlemler ana tablomda etki etmektedir.
- Viewlar normal sorgudan daha yavaş çalışır.
- Fazla sayıda view olması SQL için sağlıklı değildir.
- Sorgu uzun sürerse view'ın içine bakarız ve scriptleri hangi sorgunun yavaşlattığını bulmak için silerek deneriz
- View desing kısmında detaylı bir şekilde ayarlama yaparım.
- **View'ın New view(sihirbaz kullanımı) ile ilgili tablolarda karmaşık sorguları çoklu joinli sorgularını kolaylıkla yapabiliriz. Where şartlarını kolaylıkla sağlayabiliriz. Sıralamalar oluşturabiliriz. Hatta karmaşık join sorgularını bu pencereden bile yapabiliriz.**

```
CREATE VIEW VW_CUS
AS
SELECT * FROM CUSTOMER WHERE CITY LIKE 'KARABÜK%'

SELECT * FROM DBO. VW_CUS
```

- View içerisinde order by kullanılmaz. Kullanmak istiyorsak sorgumda TOP N kullanarak sonuna order by kullanılmalıdır.

```
SELECT TOP 100 PERCENT * FROM DBO. VW_CUS ORDER BY AGE
```

View kaynak koduna ulaşmak için database adına tıklanarak view a tıklanır ve ilgili view a design'a tıklanır.

WITH ENCRYPTION KOMUTU

Yazdığımız View'ın kaynak koduna ulaşılmasını istemiyorsan bu komut kullanılır.

```
CREATE VIEW VW_CUS
WITH ENCRYPTION
AS
SELECT * FROM CUSTOMER WHERE CITY LIKE 'KARABÜK%'
```

WITH SCHEMABINDING KOMUTU

Yazdığımız View'ın tabloda fiziksel değişiklikleri engeller.

```
CREATE VIEW VW_CUS
WITH SCHEMABINDING
AS
SELECT * FROM CUSTOMER WHERE CITY LIKE 'KARABÜK%'
```

WITH CHECK OPTION KOMUTU

View sorgusundaki where şartına göre kayıt ekletir eğer şarta uymuyorsa kayıt ekleyemez.

```
CREATE VIEW VW_CUS
AS
SELECT * FROM CUSTOMER WHERE CITY LIKE 'KARABÜK%'
WITH CHECK OPTION
```

SP_HELPTEXT VİEW IN QUERYSİNİ GÖRMEMİZİ SAĞLAR: exec sp_helptext'vw_sergi'

FONKSİYONLAR

- Fonksiyon bir veya daha fazla işlem satırından oluşan kodların bir kod bloğu şeklinde bir isim altında toplanması olarak ifade edilebilir. Böylece, sadece fonksiyon ismi çağrılarak, fonksiyon içinde yer alan kodlar çalıştırılır.

- İki çeşit fonksiyon vardır. Scalar ve Inline fonksiyonlarıdır. Scalar fonksiyonlar geriye istediğimiz tipte değer gönderir. Inline fonksiyonlar geriye tablo döndürür.

- Programability -> functions'ın altında scalar ve table olarak fonksiyonlarımızı görebiliriz.

- Herhangi bir kolonda fonksiyon kullanarak otomatik hesaplanabilir kolonlar oluşturmak mümkündür.

SCALAR FONKSİYONLAR

```
CREATE FUNCTION TOPLA. ÇARPIM(@SATIŞADEDİ INT ,@SATIŞTUTARI INT ) RETURNS INT(SCALAR
OLDUĞU GERİYE DÖNDÜRMESİNİ İSTEDİĞİMİZ DEĞER INT GELİR)
AS
BEGIN
RETURN @SATIŞADEDİ*@SATIŞTUTARI
END
```

Bu şemayı çağırırken Dbo. topla.çarpım olarak çağırabiliriz.

```
--KULLANIMI
SELECT URUNADI, DBO. TOPLA. ÇARPIM(SATIŞADEDİ, SATIŞTUTARI) AS HESAPLANMIŞ FROM URUNLER
```

INLINE FONKSİYONLAR(TABLE FONKSİYONLAR)

View ile yapılan tüm işlemler inline ile de yapılabilir. Ancak view daha kullanışlıdır.

```
CREATE FUNCTION FC_GONDER(@AD NVARCHAR(MAX),@SOYAD NVARCHAR(MAX)) RETURNS TABLE
AS
RETURN SELECT ADİ, SOYADİ FROM PERSONELLER WHERE ADİ = @AD AND SOYADİ=@SOYAD

--KULLANIMI
SELECT * FROM DBO. FC_GONDER('NANCY','DAVOLİO')
```

FONKSİYONLARDA WITH ENCRYPTION KOMUTU

Yazmış olduğum fonksiyonların kodlarına erişimi engeller.
As den önce gelir.

```
CREATE FUNCTION ORNEK(@A INT) RETURNS TABLE
WITH ENCRYPTION
AS
RETURN SELECT * FROM CUSTOMER WHERE AGE > @A
```

- **RETURN GERİYE TEK DEĞER DÖNDÜRMEDE YARAR, RETURNS GERİYE GÖNDÜRÜLECEK VERİNİN TİPİNİ BELİRLER.**

```
--KDV HESAPLAMA FONKSİYONU

CREATE FUNCTION KDV(@FIYAT DECIMAL(18,2))
RETURNS DECIMAL(18,2)
AS
BEGIN
RETURN  @FIYAT*18
END
SELECT AD, SATISFIYATI, DBO. KDV(SATISFIYATI) AS 'KDV Lİ' FROM TBLURUN
```

STORED PROCEDURE

- Normal sorgulardan daha hızlı sorgu yapmamızı sağlar. Güvenlidir kritik raporlar için sp bazında yetkiler verir.
- Ram hızında çalışırız. Vıew a göre daha güvenlidir.
- Select, Delete, Update, İnsert işlemleri yapabiliriz. TSQL in tüm komutlarını yazabiliriz.
- Execute plan işlemi yapılır. Exec komutu ile çağırırız.
- SQL injection saldırılarına karşı koruyabilir.
- Programmability -> stored procedurs de bulunur
- Stored procedureleri birbirleri içerisinde çağırabiliriz.
- Stored procedurelerin değişkenleri aksi söylenmedikçe inputtur.
- SP_SPACEISER tablonun boyutunu söyler. **SP_SPACEUSED CUSTOMER**

STORE PROCEDURE OLUŞTURMA

```
CREATE PROC SP_AD
@AD NVARCHAR(MAX)
AS
SELECT * FROM CUSTOMER WHERE NAMESURNAME LIKE @AD
--KULLANIMI
EXEC SP_AD 'S%'
```

SP İLE DEĞİŞKEN ATAMALI ÖRNEK

```
CREATE PROC SP_AMÜŞTERİLER
(@A NVARCHAR(MAX),@B INT,@C NVARCHAR(MAX))
AS
INSERT CUSTOMER (NAMESURNAME, AGE, CITY) VALUES(@A,@B,@C)
SELECT * FROM CUSTOMER
PRINT @A+' '+SUSER_NAME()+' TARAFINDAN ' + CONVERT(NVARCHAR(MAX),GETDATE()) + ' TARİHİNDE
EKLENDİ.'
INSERT INTO RAPOR(RAPOR) VALUES (@A+' '+SUSER_NAME()+' TARAFINDAN ' +
CONVERT(NVARCHAR(MAX),GETDATE()) + ' TARİHİNDE EKLENDİ.')
SELECT * FROM RAPOR
EXEC SP_AMÜŞTERİLER 'KALENDER BÜYÜK',27,'KARAMAN
```

SP DE OUTPUT İLE DEĞER DÖNDÜRME

Output parametresi dışardan atanan değişkendir içerdeki değerleri alamaz bir işlemin sonunca gelmektedir.

```
CREATE PROC SP_ORNEK
@ID INT
@ADI NVARCHAR(MAX) OUTPUT
@SOYADI NVARCHAR(MAX) OUTPUT
AS
SELECT @ADI=ADI,@SOYADI=SOYADI FROM PERSONELLER
WHERE PERSONELID=@ID
```

OUTPUT PARAMETRE TANIMLAMASI

```
CREATE PROC ORTALAMA
(@A INT , @B INT , @C INT ,@S INT OUTPUT)
AS
SET @S=(@A+@B+@C)/3
PRINT 'ORTALAMA :' + CONVERT(NVARCHAR(MAX), @S)
DECLARE @ORT INT
EXEC ORTALAMA 4,5,6,@ORT
```

TRANSACTION

Birden çok işlemin bir arada yapıldığı durumlarda eğer parçayı oluşturan işlemlerden herhangi birinde sorun olursa **tüm işlemi iptal eder. Ya da sorunlu işlemi atlayarak devam eder.**

Begin Tran İşlemi başlatır.
Commit ya da **Commit tran** işlemi başarıyla sona erer. Eğer hata alıyorsa tüm işlemlere **Rollback** ile başa sarar.
Rollback Tüm işlemleri geri alır.

DBCC OPENTRAN -> ETKİN TRANSACTIONLARI GÖSTERİR.

TRANSACTION YAZIMI:

```
DECLARE @X INT
DELETE FROM CUSTOMERYEDEK WHERE ID>1
SET @X = @@ROWCOUNT
IF @X > 1
BEGIN
PRINT 'KAYIT SİLİNDİ'
COMMIT
END
ELSE
BEGIN
PRINT 'İŞLEMLER GERİ ALINDI'
ROLLBACK
END
```

SP+BEGIN TRAN İLE BANKALAR ARASI HAVALE İŞLEMLERİ

```
CREATE PROC SP_BANKA
(       @BANKAKIMDEN NVARCHAR(MAX),
        @GONDERENHESAPNO INT,
   @ALANHESAPNO INT,
   @TUTAR MONEY              )
AS
BEGIN TRAN BANKAKONTROL
DECLARE @ABAKIYE INT , @BBAKIYE INT, @HESAPTAKİPARA MONEY
IF  @BANKAKIMDEN = 'ABANKA'
  BEGIN
   SELECT @HESAPTAKİPARA= BAKIYE FROM ABANKA WHERE HESAPNO= @GONDERENHESAPNO
 IF @TUTAR > @HESAPTAKİPARA
  BEGIN
   PRINT CONVERT(NVARCHAR(MAX),@GONDERENHESAPNO)+' NUMARALI HESAPTA, GONDERILMEK ISTENEN TUTARDAN AZ PARA MEVCUTTUR'
    ROLLBACK
END
ELSE
   BEGIN
     UPDATE ABANKA SET BAKIYE=BAKIYE - @TUTAR WHERE HESAPNO = @GONDERENHESAPNO
  UPDATE BBANKA SET BAKIYE=BAKIYE + @TUTAR WHERE HESAPNO = @ALANHESAPNO
PRINT 'ABANKASIDANKİ ' + CONVERT(NVARCHAR(MAX),@GONDERENHESAPNO) + ' NUMARALI KALAN BAKIYE : ' + CONVERT(NVARCHAR(MAX) , @BBAKIYE)
PRINT ' SON DEĞERLER ;'
END
END
ELSE
  BEGIN
   SELECT @HESAPTAKİPARA= BAKIYE FROM BBANKA WHERE HESAPNO= @GONDERENHESAPNO
    IF @TUTAR > @HESAPTAKİPARA
  BEGIN
    PRINT CONVERT(NVARCHAR(MAX),@GONDERENHESAPNO)+ ' NUMARALI HESABA, GONDERILMEK ISTENEN TUTARDAN AZ PARA MEVCUTTUR'
SELECT @ABAKIYE= BAKIYE FROM BBANKA WHERE HESAPNO = @GONDERENHESAPNO
SELECT @BBAKIYE= BAKIYE FROM ABANKA WHERE HESAPNO = @ALANHESAPNO
PRINT 'A BANKASINDAKİ ' + CONVERT(NVARCHAR(MAX),@GONDERENHESAPNO)+ ' NUMARALI HESAPTA KALAN BAKIYE : '+ CONVERT(NVARCHAR(MAX),@ABAKIYE)
PRINT 'B BANKASINDAKİ ' + CONVERT(NVARCHAR(MAX),@ALANHESAPNO)+ ' NUMARALI HESAPTA KALAN BAKIYE : ' +  CONVERT(NVARCHAR(MAX),@BBAKIYE)
COMMIT
END
ELSE
  BEGIN
     SELECT @HESAPTAKİPARA= BAKIYE FROM BBANKA WHERE HESAPNO= @GONDERENHESAPNO
 IF @TUTAR > @HESAPTAKİPARA
  BEGIN
   PRINT CONVERT(NVARCHAR(MAX),@GONDERENHESAPNO)+' NUMARALI HESAPTA, GONDERILMEK ISTENEN TUTARDAN AZ PARA MEVCUTTUR'
    ROLLBACK
END
ELSE
   BEGIN
     UPDATE BBANKA SET BAKIYE=BAKIYE - @TUTAR WHERE HESAPNO = @GONDERENHESAPNO
  UPDATE ABANKA SET BAKIYE=BAKIYE + @TUTAR WHERE HESAPNO = @ALANHESAPNO
PRINT 'BBANKASIDANKİ ' + CONVERT(NVARCHAR(MAX),@GONDERENHESAPNO) + ' NUMARALI HESAPTAN ABANKASINDAKİ : ' + CONVERT(NVARCHAR(MAX) , @ALANHESAPNO)
+' NUMARALI HESABA' + CONVERT(NVARCHAR(MAX),@TUTAR) + 'DEĞERİNDE PARA GÖNDERİLMİŞTİR'
PRINT ' SON DEĞERLER ;'

SELECT @BBAKIYE= BAKIYE FROM BBANKA WHERE HESAPNO = @GONDERENHESAPNO
SELECT @ABAKIYE= BAKIYE FROM ABANKA WHERE HESAPNO = @ALANHESAPNO
PRINT 'A BANKASINDAKİ ' + CONVERT(NVARCHAR(MAX),@GONDERENHESAPNO)+ ' NUMARALI HESAPTA KALAN BAKIYE : ' + CONVERT(NVARCHAR(MAX),@ABAKIYE)
PRINT 'B BANKASINDAKİ ' + CONVERT(NVARCHAR(MAX),@ALANHESAPNO)+ ' NUMARALI HESAPTA KALAN BAKIYE : ' + CONVERT(NVARCHAR(MAX),@BBAKIYE)
COMMIT
END
END

END
```

TRIGGERS(TETİKLEYİCİLER)

Tetikleyici (Trigger) yapısı, ilişkisel veri tabanı yönetim sistemlerinde, bir tabloda belirli olaylar meydana geldiğinde veya gelmeden önce otomatik olarak çalışan özel bir store procedure türüdür. Bir tabloda ekleme, güncelleme ve silme işlemlerinden biri gerçekleştiğinde veya gerçekleşmeden önce, aynı tabloda veya başka bir tabloda belirli işlemlerin yapılmasını istediğimizde, trigger yapısını kullanırız. Örnek verecek olursak, satış tablosunda satış işlemi gerçekleştiğinde ürünün stok miktarının eksiltilmesi, banka hesabında işlem gerçekleştikten sonra otomatik olarak email gönderilmesi gibi örnekler verilebilir.

DML TRIGGER

- INSERT, UPDATE ve DELETE gibi veri okuma ve işleme için kullanılan DML ifadeleridir. DML tetikleyicileri, INSERT, UPDATE ve DELETE olayları kullanılarak veriler her değiştirildiğinde tetiklenir. DML tetikleyicileri iki tip olarak sınıflandırılır.
- Commit işlemi işlemler doğruysa işlemi başarılı sonlandırır. Eğer bir hata alınırsa Rollback yapar ve tüm işlemleri geriye sardırarak başa gelir.

DDL TRIGGER

- CREATE, ALTER, DROP gibi veritabanı ve tablo ile ilgili işlemler DDL ifadeleridir. DDL tetikleyicisinin amacı, bir DDL nesnesi oluşturmaya (create), değiştirmeye (alter) veya düşürmeye (drop) çalışan kullanıcılara kısıtlama getirilmesini sağlamaktadır. CREATE, ALTER ve DROP ifadeleri kullanıldığında devreye giren tetikleyici türüdür

INSERTED TABLE - DELETED TABLE

Tabloda Insert işlemi yapılıyorsa arka planda işlemler RAM'de ınserted adlı tablo eklenir. İnserted tablosundaki veriler fiziksel tabloya insert edilir ve ramde inserted tablomdaki veriler silinir. Eğer işlemim delete ise aynı işlemler deleted tablosunda izlenir.

Update işleminin mantığı ilk deleted tablosu daha sonra inserted tablosunda olur.

DML TRIGGER OLUŞTURMA

```
CREATE TRIGGER DMLTRG
ON TABLENAME
AFTER VEYA FOR DELETE, UPDATE, INSERT
AS
PRINT 'BU İŞLEM YAPILAMAZ '
ROLLBACK
```

DDL TRIGGER OLUŞTURMA

```
CREATE TRIGGER DDLTRG
ON DATABASE
AFTER VEYA FOR DROP_TABLE, ALTER_TABLE, CREATE_TABLE
AS
PRINT 'BU İŞLEM YAPILAMAZ '
ROLLBACK
```

DML TRIGGER İLE RAPOR YAZMA - DELETE

```
CREATE TRIGGER TRIGGERSILINME
ON CUSTOMER
AFTER DELETE
AS
DECLARE @ADI NVARCHAR(MAX)
SELECT  @ADI = NAMESURNAME FROM DELETED
INSERT RAPOR(RAPOR) VALUES ('ADI VE SOYADI'+' ' + @ADI + 'OLAN PERSONEL ' + SUSER_NAME() + ' TARAFINDAN '
+CONVERT(NVARCHAR(MAX),GETDATE())
+ 'TARİHİNDE SİLİNMİŞTİR')
SELECT * FROM RAPOR
```

DML TRIGGER İLE RAPOR YAZMA - UPDATE

```
UPDATE CREATE TRIGGER TG_UPDATECUSTOMER
ON CUSTOMER
AFTER UPDATE
AS
DECLARE @ESKIADI NVARCHAR(MAX) , @ADI NVARCHAR(MAX)
SELECT  @ESKIADI = NAMESURNAME FROM DELETED
SELECT @ADI= NAMESURNAME FROM INSERTED
INSERT RAPOR(RAPOR) VALUES ('ADI VE SOYADI'+' ' + @ESKIADI + 'OLAN PERSONEL ' +@ADI + ' YENİ ADIYLA
DEĞİŞTİRİLECEK '
+ SUSER_NAME() + ' TARAFINDAN ' +CONVERT(NVARCHAR(MAX),GETDATE())
+ 'TARİHİNDE GUNCELLENMISTIR')
SELECT * FROM RAPOR
```

MULTIPLE ACTIONS TRIGGER

Bir trigger üzerinden birden fazla fonksiyonu bir arada yapma işlemidir. Ancak ayrı ayrı yapmak daha mantıklıdır.

```
CREATE TRIGGER MULTITRIGGER
ON PERSONELLER
AFTER DELETE, INSERT
AS
PRINT 'BAŞARILI'
INSERT PERSONELLER(ADI, SOYADI) VALUES ('SERKAN','COŞKUN')
DELETE FROM PERSONELLER WHERE ID=131
```

INSTEAD OF TRIGGER İLE RAPOR YAZMA UPDATE,

Bu trigger ı atadığımızda insert, update, delete işlemlerini yaptırmaz yerine belirlediğimiz işlemi yapar.

Customer tablosunda update gerçekleştiği anda yapılacak güncelleştirme yerine bir rapor tablosuna
('ADI VE SOYADI ' + @ADI+' OLAN KULLANICI ' + SUSER_NAME()+ ' TARAFINDAN '+
CONVERT(NVARCHAR(MAX),GETDATE())+'TARİHİNDE DEĞİŞTİRİLMEK İSTENDİ.') kalıbında bir rapor yazan trigger yazalım.

```
CREATE TRIGGER INSTEADTRIGGER
ON CUSTOMER
INSTEAD OF (UPDATE),DELETE, INSERT  ---> uptade yerine anlamına gelir
AS
DECLARE @ADI NVARCHAR(MAX)
SELECT @ADI=NAMESURNAME FROM deleted
SELECT @ADI=NAMESURNAME FROM inserted
INSERT RAPOR(RAPOR) VALUES ( 'ADI VE SOYADI ' + @ADI+' OLAN KULLANICI ' + SUSER_NAME()+ ' TARAFINDAN '+
CONVERT(NVARCHAR(MAX),GETDATE())+
'TARİHİNDE DEĞİŞTİRİLMEK İSTENDİ.')
```

SP İLE DEĞER EKLEME VE GUIDLI DEĞERE TRIGGER İLE DEĞER ATAMA

```
CREATE PROC SP_GİRİLMEYEN_DEĞERLER
( @AD NVARCHAR(MAX) = 'İSİM GİRİLMEDİ',
  @CITY NVARCHAR(MAX) = 'ŞEHİR GİRİLMEDİ',
  @YAS NVARCHAR(MAX) = 'YAS GİRİLMEDİ',
  @YASLI NVARCHAR(MAX) = 'YASLILIK GİRİLMEDİ')
  AS
  INSERT CUSTOMER(NAMESURNAME, CITY, YAS, YASLI) VALUES (@AD,@CITY,@YAS,@YASLI)
CREATE TRIGGER OZELDEGER
ON CUSTOMER
AFTER INSERT
AS
DECLARE @OZEL uniqueidentifier = NEWID()
UPDATE CUSTOMER SET OZEL=@OZEL WHERE id = @@IDENTITY
  EXEC SP_GİRİLMEYEN_DEĞERLER 'KAHRAMAN DENİZ COŞKUN','ANKARA'
```

INDEXLER

- Dikeyde çalışmamızı sağlarlar.
- Bir telefon rehberinde düzensiz halde 1000 kişi varsa SQL aramada 1000 pagelik sorgu gelir ancak indexlerin ikiye böl bul metoduyla 1000 pagelik sorguyu 9, 2000 i 10 adımda bulur 1 milyar pagelik değeri 29 adımda bulabilir. Buda sorguyu hızlı çekmemi sağlar. İndexler bu telefon rehberinde sıralama yapmayı sağlar.
- Bir page 8kbdır.
- Karmaşık sorgumu çalıştırdığımda ve CTRL+L ile execution plan penceresi gelir. Burada işlem okumasını neyle yaptığını söyler.
  Table scan ile okumuş olması sorguyu teker teker okuduğu anlamına gelmektedir.
  Clustred index scan ise index ile okuma yaptığı anlamına gelmektedir.
- Database -> Tables -> ilgili tablo başlığının altında indeksleri görebiliriz.
- Karmaşık sorgularda içinde Where ve order by olan sorgularda. Execution planı açtığımızda sağ tık yaptığımızda missing index(ınpact.) olarak bize en uygun index yapısını verir.

  Where şartım hangi kolona etki ediyorsa o kolonu indexlememiz lazımdır.
- Üç çeşit index vardır **Clustered index, nonclustered index, unique index**
- Clustered index primary key olan indextir tablomda sadece 1 adet olabilir.
- Büyük databaselerde örneğin barkod okutma sistemli bir firmada indexlerin ikiye böl bul metodu ile saniyeler içinde sorguma ulaşabilirim.
- SP_UPDATESTATS ile index frekanslarına bakarım index frekansı kaç tane olduğunu söyler.
- İndexlerin fragmantasyonları %70 üstü olmalıdır.

INDEX OLUŞTURMA

```
CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]ON [dbo].[USER_]
([NAMESURNAME])
CREATE CLUSTERED INDEX CUSTOMER ON CUSTOMER(ID)
```

NON CLUSTERED İNDEXLER

NON CLUSTERED İNDEXLER HAKKINDA BİLGİ VERİYOR
https://www.ismailgursoy.com.tr/non-clustered-index-nedir/

INDEX OLUŞTURMA VE SİLME-CLUSTREDED VE UNIQUE INDEX OLUŞTURMA

```
CREATE INDEX IX_CS_CITY ON CUSTOMER (CITY DESC);
DROP INDEX IX_CS_CITY ON CUSTOMER
ALTER INDEX IX_CS_CITY ON CUSTOMER DISABLE or REBUILD Pasif hale getirir.
CREATE CLUSTERED INDEX CS_PKEY ON CUSTOMER(ID)
CREATE UNIQUE INDEX CS_UNIQ ON CUSTOMER (OZEL)
```

INDEXLER

https://medium.com/trendyol-tech/sql-server-index-yap%C4%B1s%C4%B1-aba652828fe2

https://www.ismailgursoy.com.tr/index-nedir-ne-ise-yarar/

INDEX BAKIMI TOPLU REBUILD

Managemnet -> Maintance Plans -> New Maintance -> Name: Index Rebuild

Sol kenardan Toolboxtan -> rebuilld index task a çift tıklarım düzletecek indexleri seçerim.

Küçük kareli pencerede işaretinden zaman sıklıklarını seçeriz.

Fragmantasyonlar indexlerin ne kadar bozuldukları bilgisini verirler.

İSTATİSTİK GÜNCELLEME

A normal durumlar yaşandığında yapılır.

Maintance Plan -> New Main ->Name: İstatistiksel güncelleme

Toolbox -> update statistic tase üzerinde hangi dosyalarda yapacağımı seçerim.

Küçük kareli pencerede işaretinden zaman sıklıklarını seçeriz.

UPDATE STATISTICS USER_ İLE TABLO BAZLI İSTATİSTİK GÜNCELLEYEBİLİRİM.

JOB

BELİRLEMİŞ OLDUĞUMUZ BİR TAKIM İŞLERİ İSTEDİĞİMİZ ARALIKLARDA YAPAN KOMUT KÜMELERİDİR.

JOB SAYESİNDE DÜZENLİ YEDEKLEMELER ALINABİLİR.

JOB OLUŞTURMA:

SQL SERVER AGENT -> sağ klik ile start a tıklarız-> Yes ve artık aktifleştirdik.

Database de Etrade databasesinde JOBS adında bir tablo oluştururum ve ID ve Metin adında kolonlar oluştururum.

Örneğin yapacağım her yedeklemede bana zaman bilgisi ve başarılı bilgisi vermesini isterim.

SQL SERVER AGENT->JOBS->NEW JOB -> JOB 1 ADI VERİRİM -> SOL SÜTUN BAŞLIKLARINDAN STEPS E TIKLARIM.

NEW-> STEPNAME VERİRİM(DENEME)-> DATABASES KISMINDAN İLGİLİ DATABASE İ SEÇERİM. COMMAND KISMINA

OLUŞTURDUĞUM TABLOYA İNSERT EDECEK DEĞERİ YAZARIM.

```
INSERT JOBS(METIN) VALUES ('BAŞARILI'+ CONVERT(NVARCHAR(MAX),GETDATE()))
```

OK DERİZ

SOL SÜTUNDAN SCHEDULES(ZAMANLAMA AYARLAMALARI) E TIKLARIM.

NEW->NAME GİRİLİR ->FREQUENCY DEN ZAMANLAMA SEÇİLİR. DAİLY İ SEÇTİM -> OCCURS EVERY DEN SAATTE

DAKİKADA YA DA SANİYEDE KAÇ DEFA YAPILMASI GEREKTİĞİNİ AYARLARIZ. VE DİĞER TÜM İŞLEMLERE OK DİYEREK

JOBIMIZI OLUŞTURURUZ.

EĞER JOB KLASÖRÜNDE DEĞİŞİKLİK YA DA SİLME İŞLEMİ YAPMAK İSTİYORSAK JOBS + YA TIKLAYARAK İLGİLİ JOB I

SEÇERİZ VE DÜZENLEMELERİ YAPARIZ.

KARIŞIK SQL BİLGİLERİ - TSQL SORGULARI

DBCC CHECKDB İLE DATA BASE SAĞLAMLIK KONTROLÜ

```
DBCC CHECKDB ('DBBANK')
```

BOZUK PAGE TESPİT ETME

DB SAĞTIK -> TASKS -> RESTORE -> PAGE -> CHECK DATABASES buradan bozuk olan sayfalar gelir.

WAİTFOR DELAY VE TİME

Sorguyu istenilen süre sonra çalıştırır, time ise verilen zamanda çalıştırır.

```
WAITFOR DELAY '00:00:05'
SELECT * FROM CUSTOMER

WAITFOR TIME '14:12:00'
SELECT * FROM CUSTOMER
```

BULK INSERT

Harici bir kaynaktan veri ekleme. Önceden fiziksel bir tablom olmalıdır!

```
BULK INSERT KISILER
FROM 'D:\KİSİLER. TXT'
WITH /*BULK INSERT E ÖZEL BİR WITH*/
(FIELDTERMINATOR ='\T' -> SÜTUN İÇİN
 ROWTERMINATOR ='\N' -> SATIR İÇİN )
```

VERİ TABANINDAKİ TABLOLARI LİSTELEME

```
SELECT * FROM SYS. TABLES
```

VERİ TABANINDAKİ DİLİ ÖĞRENME

```
SELECT  @@LANGUAGE
```

SQL VERSİYONUNU ÖĞRENME

```
SELECT  @@VERSION
```

LİNKED SERVER

Bu işlem uzak sunucu bağlantılarını tek çatı altında kendi PC'mi de toplamamı sağlar.

Server Object -> New Server -> Sql Server ı seçerim

General Dent -> Linked Serverdan -> Ip adresi

Security -> Local Login = BİADMİN remoteover bi admin şifresi girilir.

Bc mode using… I seçerim

Linked Server dosyasına IP adresi ile gelir.

DATABASE VE LOG DOSYASI OLUŞTURMA

```
CREATE DATABASE SERKAN ON ( NAME ='SS'(ARKA PLAN İSMİ) ,
FILENAME = 'D:/SS. MDF' (FİZİKSEL DİZİLİMİ ARKA PLAN İÇİN),
SIZE = 3 (BAŞLANGIÇ BOYUTU),
FILEGROWTH = 3(BÜYÜME BOYUTU))
LOG ON
( NAME ='SS',  FILENAME = 'D:/SS. MDF',
SIZE = 3, FILEGROWTH = 3)
```

SQL DATABASELER

- MASTER DB: Master database, Sistem üzerinde kullanıcıların açmış olduğu database'lerin listesini, yine kullanıcı tarafından veya sistem tarafından belirlenen login bilgileri gibi birçok veriyi saklar ve yönetir.

  Master kopyalanıp şifresi değiştirilirse diğer kullanıcılar eski şifreyle giremez. Bu işlem için SQL in kapalı olması gerekir.

- MODEL DB: Boş DB dir yeni oluşturulan DB'lerin veri tablolarını almaktadır.

  Yeni db oluşmuyorsa bunun sebebi modelin açık olmasıdır.

- MSDB: Periyodik olarak çalıştırılan her türlü işlemler (joblar,schedulelar,alerttes) burada tutar.

  Job: Periyodik olarak yapılan işlemler shedulelar periyotlardır.

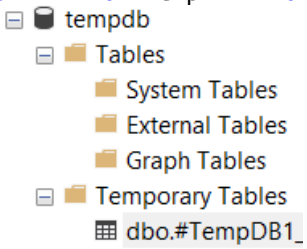
  SQL server agent: Job : Günde 10 kez tarih geçir .

  New Job -> ad verilir. Steps -> steps name -> Database seçilir ve altına ilgili komut yazılır. Ok denir.

  SCHEDULE -> name -> altında istenilen aralık tanımlanmış jobları aktarmak istiyorsam MSBD'yi kopyalamam yeterli ancak SQL versiyonlarım aynı olmak şartı vardır.

- TEMP DB: Geçici Tablolar bu db de tutulur. Aggregate fonksiyonları, join ile olan tablolar arka planda bunu yapan tablolar kullanılır # ve ##

```
SELECT * INTO #TempDB1 FROM CUSTOMER
```



SQL YEDEK ALMA

BACKUP İLE YEDEK ALMA

İlgili database seçilir. Tasks -> Backup -> Varsayılan C ye aktar Ok ile alınır.

SCRİPT İLE YEDEK ALMA

İlgili database seçilir. Tasks -> Generate scripts -> next -> select specifics … Seçilir -> kutucuklardan sadece istenilen başlıkların yedekleri seçilir (Tables)

Advancedd -> typses of data to script kısmında schema and data seçilir. Üç noktayı … Seçerek kaydolacak yeri ayarlarız ve finishleriz.

LOG DOSYASI İLE YEDEK AÇMA

DataBases sağ tık -> attach -> add -> log (mdf ya da ldf) dosyası seçilerek içeri aktarılır.

SQL EXPRESS SÜRÜMÜNDE OTOMATİK YEDEK ALMA İŞLEMLERİ

SQL Server'ın ücretsiz sürümü olan Express edition birçok yerde malum sebeplerden tercih ediliyor. Bu sürümün en önemli eksikliklerinden biri de SQL Agent hizmetinin çalışmıyor olması. Bilmeyenler için şöyle diyebiliriz, bu servis SQL Server 'da otomatik ve periyodik olarak çalışacak işlemleri kontrol eden servis. Otomatik ve periyodik olarak çalışan denince de akla ilk gelen işlem tabi ki sistemin yedeğinin otomatik olarak alınması.

Birçok kişi bu eksiklik sebebiyle ya sistem yedeğini hiç almıyor, ya manuel olarak alıyor, ya da Veeam tarzı uygulamalar ile günde bir kez tüm sistemin yedeğini alıyor. Oysa gün içerisinde gerçekleşebilecek bir problemde bir gün önceki yedek yeterli olmayabilir.

Şimdi hadi gelin bu sorunu çözmek için çok pratik bir işlem yapalım.

Öncelikle veri tabanımızın bir yedeğini almamız gerekiyor. Tabi birden fazla veri tabanı varsa her biri için de bu işlemin otomatik olarak gerçeklemesi gerekiyor.

Bir sonraki aşamada bu yedekleri rar ile sıkıştırıp başka bir yere de kopyalamak isteyebiliriz. Onu da sadece sql scripti kullanarak yapabiliriz. Tabi makinemizde winrar kurulu olması gerekiyor.

Aşağıdaki script işte tam olarak bu işe yarıyor. Yani otomatik olarak belirlediğimiz tüm database lerin yedeklerini alıyor. Şimdi bu scripti master db seçili iken çalıştıralım.
Tabi buradaki klasörler bilgisayarınızda yoksa oluşturmalısınız.

SQL SERVER'DA BACKUP STRATEJİŞERİ FULL BACKUP VE DIFFERENTIAL BACKUP

1- https://www.cozumpark.com/sql-serverda-backup-stratejileri-1-full-backup-ve-differential-backup/

2- https://www.cozumpark.com/sql-serverda-backup-stratejileri-2-full-backup-ve-backup-compression/

3- https://www.cozumpark.com/sql-serverda-backup-stratejileri-3-differential-backup-kavrami-ve-kullanimi/

4- https://www.cozumpark.com/sql-serverda-backup-stratejileri-4-transaction-log-backup/

5- https://www.cozumpark.com/sql-server-backup-stratejileri-5-maintenance-plan-ile-otomatik-backup-plani-olusturma/

SQL SERVER'DA CANLI VERİYİ TABLO BAZLI SIKIŞTIRMA (COMPRESS)

SP_SPACEUSED 'TABLO ADI' ile tablo boyutumun bilgilerini almaktayım

Bu yazımızda SQL Server'daki bir tabloyu sıkıştırma yani Compress özelliğinden bahsediyor olacağım. Birçoğumuz veri tabanlarında text veriler kullanıyoruz. Bu verilerde ise gerek veri tabanı mimarisi sebebiyle ya da gerekse içerisindeki veriler sebebiyle boşluklar bulunmakta. Bu boşluklar ise gereksiz yer teşkil etmekte.

Tablo-> sağ tık ile storage -> manage compression -> next -> row -> calculate -> tekrar page seçilir -> calculate burada ne kadar sıkışacağı bilgisini veriyor. -> next dediğimde sorguyu veriyor

```
USE [ETRADE2]
ALTER TABLE [dbo].[SALESA] REBUILD PARTITION = ALL
WITH
(DATA_COMPRESSION = PAGE)
```

- Bu yazıda SQL Server'da compression özelliğini ve nasıl kullanılacağını anlattık.
- 2 GB lık bir database'i 85 MB'a kadar küçülttük.
- Ayrıca row based ve page based sıkıştırmanın farkını gördük.
- Son olarak compress aktif bir tabloda pasif olana göre beklediğimizin tam aksi daha az okuma ve daha fazla performans gösterdiğini gördük.

https://omercolakoglu.net/2020/09/

SQL SERVER'DA BİLİNÇLİ İNDEXLEME

Indexler yaşayan varlıklar. Yeni kayıtlar eklendiğinde, kayıtlar silindiğinde ya da güncelleme işlemi yapıldığında haliyle bu indexler bozuluyor ve belli zaman aralıklarında bunları güncellemek gerekiyor. Bunun yolu da çok basit. 2 dakikada bu işi yapacak bir planı kod yazmadan oluşturabilirsiniz.
Bu düzeltme işlemi bazen uzun sürüyor ve sistemi kilitliyor ve durdurmak zorunda kalıyorsunuz. Siz hangi tablonun indexlerinin yapıldığını hangi tablonun yapılmadığını bilemiyorsunuz.
Ayrıca hangi tablo indexten önce ne kadar bozuktu, indexten sonra ne kadar düzeldi onu da bilmiyorsunuz.
Yine hangi tablodaki index düzeltme işleminin ne kadar sürdüğünü de bilmiyorsunuz.
İşte bu sorunlara çözüm olması amacı ile bir stored procedure yazdım ve onu burada paylaşmak istiyorum.
Bunun için index işlemini loglamak amacı ile bir veri tabanı ve bir tablo oluşturuyoruz.
Örneğin ben burada database i BT diye oluşturdum.
Siz de aşağıdaki script ile oluşturabilirsiniz.

```
use BT
CREATE TABLE [dbo].[TBLINDEXLOG](
    [ID] [int] IDENTITY(1,1) NOT NULL,
    [SEQID] [int] NULL,
    [DBNAME] [varchar](50) NULL,
    [TABLENAME] [varchar](250) NULL,
    [BEGDATE] [datetime] NOT NULL,
    [ENDDATE] [datetime] NULL,
    [DURATION] [int] NULL,
    [FRAGMANTATIONBEFORE] [float] NULL,
    [FRAGMANTATIONAFTER] [nchar](10) NULL,
    [ROWCOUNT_] [int] NULL,
    [DATASIZEBEFORE] [int] NULL,
    [INDEXSIZEBEFORE] [int] NULL,
    [TABLESIZEBEFORE] [int] NULL,
    [DATASIZEAFTER] [int] NULL,
    [INDEXSIZEAFTER] [int] NULL,
    [TABLESIZEAFTER] [int] NULL,
 CONSTRAINT [PK_TBLINDEXLOG] PRIMARY KEY CLUSTERED
(
    [ID] ASC

) )

CREATE PROC [dbo].[SPREINDEX]
@TABLENAME AS VARCHAR(1000)='%',
@FILLFACTOR AS INT=70
AS
DECLARE @ID AS INT
DECLARE @SEQID AS INT
DECLARE @BEGDATE AS DATETIME
DECLARE @ENDDATE AS DATETIME
DECLARE @DURATION AS INT
DECLARE @ROWCOUNT AS INT
DECLARE @DATASIZEBEFORE INT
DECLARE @INDEXSIZEBEFORE AS INT
DECLARE @TABLESIZEBEFORE AS INT
DECLARE @DATASIZEAFTER INT
DECLARE @INDEXSIZEAFTER AS INT
DECLARE @TABLESIZEAFTER AS INT
DECLARE @FRAGMANTATIONBEFORE AS FLOAT
DECLARE @FRAGMANTATIONAFTER AS FLOAT

SELECT @SEQID=MAX(SEQID)+1 FROM BT.DBO.TBLINDEXLOG
SET @SEQID=ISNULL(@SEQID,1)

CREATE TABLE #T (NAME VARCHAR(200),ROWS INT,RESERVED VARCHAR(100),DATA VARCHAR(100),INDEX_SIZE VARCHAR(100),UNUSED
VARCHAR(100))
```

```
DECLARE @TABLENAME2 AS VARCHAR(1000)
DECLARE CRS CURSOR FOR SELECT NAME FROM SYSOBJECTS WHERE XTYPE='U' AND NAME LIKE @TABLENAME
OPEN CRS
FETCH NEXT FROM CRS INTO @TABLENAME2
WHILE @@FETCH_STATUS=0
BEGIN
	SET @BEGDATE=GETDATE()
	TRUNCATE TABLE #T
	INSERT INTO #T
	EXEC SP_SPACEUSED @TABLENAME2

	SELECT @ROWCOUNT=ROWS
	,@TABLESIZEBEFORE=CONVERT(INT,REPLACE(RESERVED,' KB',''))
	,@DATASIZEBEFORE=CONVERT(INT,REPLACE(DATA,' KB',''))
	,@INDEXSIZEBEFORE=CONVERT(INT,REPLACE(INDEX_SIZE,' KB',''))
	FROM #T

	SELECT @FRAGMANTATIONBEFORE=avg(avg_fragmentation_in_percent) FROM sys.dm_db_index_physical_stats(DB_ID(),
OBJECT_ID(@TABLENAME2), NULL, NULL , 'LIMITED')
	WHERE index_type_desc <>'CLUSTERED INDEX'


	 INSERT INTO BT.DBO.TBLINDEXLOG
	(SEQID, TABLENAME,DBNAME, BEGDATE,ROWCOUNT_,DATASIZEBEFORE,INDEXSIZEBEFORE,TABLESIZEBEFORE,FRAGMANTATIONBEFORE)
	VALUES
	(@SEQID, @TABLENAME2,DB_NAME(),
@BEGDATE,@ROWCOUNT,@DATASIZEBEFORE,@INDEXSIZEBEFORE,@TABLESIZEBEFORE,@FRAGMANTATIONBEFORE)

	SET @ID=@@IDENTITY

	DECLARE @SQL AS NVARCHAR(MAX)
	SET @SQL='ALTER INDEX ALL ON '+@TABLENAME2+' REBUILD WITH (FILLFACTOR='+CONVERT(VARCHAR,@FILLFACTOR)+')'
	EXEC SP_EXECUTESQL @SQL

	TRUNCATE TABLE #T
	INSERT INTO #T
	EXEC SP_SPACEUSED @TABLENAME2

	SELECT @ROWCOUNT=ROWS
	,@TABLESIZEAFTER=CONVERT(INT,REPLACE(RESERVED,' KB',''))
	,@DATASIZEAFTER=CONVERT(INT,REPLACE(DATA,' KB',''))
	,@INDEXSIZEAFTER=CONVERT(INT,REPLACE(INDEX_SIZE,' KB',''))
	FROM #T

	SET @ENDDATE=GETDATE()
	SET @DURATION=DATEDIFF(SECOND,@BEGDATE,@ENDDATE)

	SELECT @FRAGMANTATIONAFTER=avg(avg_fragmentation_in_percent) FROM sys.dm_db_index_physical_stats(DB_ID(),
OBJECT_ID(@TABLENAME2), NULL, NULL , 'LIMITED')
	WHERE index_type_desc <>'CLUSTERED INDEX'

	SET @SQL='UPDATE STATISTICS '+@TABLENAME2
	EXEC SP_EXECUTESQL @SQL

	UPDATE BT.DBO.TBLINDEXLOG SET
	ENDDATE=@ENDDATE, DURATION=@DURATION, DATASIZEAFTER=@DATASIZEAFTER, INDEXSIZEAFTER=@INDEXSIZEAFTER,
	TABLESIZEAFTER=@TABLESIZEAFTER,
	FRAGMANTATIONAFTER=@FRAGMANTATIONAFTER
	WHERE ID=@ID


FETCH NEXT FROM CRS INTO @TABLENAME2
END
CLOSE CRS
DEALLOCATE CRS

EXEC DBO.SPREINDEX

SELECT * FROM BT.DBO.TBLINDEXLOG ORDER BY FRAGMANTATIONBEFORE DESC
```

Çıkan sonuç: en çok satır sayıları olan tabloları incelememizi sağlıyor. Örneğin burada LG_217_01_STLINE tablosunu inceleyelim
%31 fragmante olmuş yani indexler %31 oranında bozuk.
159.469 satır var
Index süresi 42 Sn. sürmüş.
Tablo boyutu indexten sonra 409.368 KB'tan 492.056'a çıkmış. Bu büyüme Fill factor değerinden kaynaklanıyor.

SQL SERVER DATABASE MAİL VE GMAİL İLE KULLANIMI

```
EXEC MSDB.DBO.SP_SEND_DBMAİL
@PROFİLE_NAME='SQL MAİL'
@RECİPİENTS = 'serkancoskun@aktekltd.com'
@BODY = 'Bu bir test mailidir.'
@SUBJECT= 'TEST MAİL'
EXECUTE İLE SQL SERVER BİZE MAİL GÖNDEREBİLİR.
```

https://www.cozumpark.com/sql-server-database-mail-ve-gmail-ile-kullanimi/

SQL SERVER ' I RESTART ETMEK DOĞRU BİR DAVRANIŞ MI ?

Ancak, birkaç durumda, bunu yapmak zorundayız:

- Windows bakımı;
- Ağ protokollerinin yapılandırmasını etkinleştirme/devre dışı bırakma veya değiştirme (SQL Server Configuration Manager kullanarak);
- Düzeltmeler/Yamalar/Hizmet paketlerinin uygulanması;
- Dosyanın yeniden oluşturulmasıyla giderilen tempdb sorunları;
- İzleme bayrakları gibi başlangıç parametrelerinin uygulanması;
- Sunucu kimlik doğrulama modu değişiklikleri.

Yukarıdaki nedenler dışında, bakımlı ve sağlıklı bir SQL Server sonsuza kadar mutlu bir şekilde çevrimiçi kalabilir.

Bana göre, sunucunun yeniden başlatılması muhtemelen performansa en çok zarar veren şeylerden biri! SQL sunucusunu yeniden başlattığımızda, tüm belleği sunucu işletim sistemine geri alıyoruz, plan önbelleğini tamamen temizliyoruz ve tüm tempdb'leri siliyoruz. Bu şu anlama gelir:

- Belleği Windows işletim sisteminden geri almak biraz zaman alacaktır. Bu süre boyunca, SQL sunucusu diskten çok fazla veri okuyacak ve verileri belleğe geri yükleyecektir.
- SQL Server'ın Plan Önbelleği, sıklıkla yürütülen sorgular için önceden derlenmiş yürütme planını saklar. Yürütme planlarını oluşturma ve tekrar tekrar derleme maliyetini azaltarak sorgu performansını artırır.
- Önbelleğe alınmış tüm sorgu planları kaybolur ve SQL sunucusunun onu yeniden derlemesi gerekir. Sunucunun iş yüküne bağlı olarak, SQL sunucusunun tüm yürütme planlarını oluşturup önbelleğe geri yüklemesi biraz zaman alacaktır. "Yeni Plan Önbelleği", muhtemelen nadiren yürütülen prosedürleri içermeyecektir, çünkü hizmet sona erdiği için tüm planlar "anında" oluşturulacaktır. Ek olarak, kötü bir yürütme planı "alma" riskini alıyoruz.

https://omercolakoglu.net/2018/07/03/sql-serveri-restart-etmek-dogru-bir-davranis-mi/

SQL SERVER'DA SUSPECT MODE'A DÜŞEN BİR DATABASE KURTARMA

https://omercolakoglu.net/2018/07/

DONANIMLARIN SQL SERVER PERFORMANSI ÜZERİNE ETKİSİ

SQL DE GEÇ KALAN VERİNİN NEDENİ

- Sorgumun geç gelmesinin ana sebebi ramden kaynaklı olabilir . Görev yöneticisinden CPU 'mun ve AVG min yorulmasını görebilirim.
- CPU değerimin 20-25 olmasını beklerim. Ram büyüklüğü sorgu çekmemi kolaylaştırır.
- Sql serverda ram de yer kalmazsa devinim söz konusudur.
- Ram in bus hızı önemlidir. Doğrusal anlamda etkilidir. Ancak sadece ram in yüksek olması bir çözüm değildir.
- Çekirdek sayısı sorguyu kolaylaştırır.
- Join işleminde tablo kullanımı için birer çekirdeği meşgul eder.

https://www.youtube.com/watch?v=TJj6-br8xh0&t=283s

SQL DATABASELERİNİN AYLIK NE KADAR BÜYÜDÜĞÜNÜ GÖSTEREN SORGU

İleriye yönelik sistem kaynak planlaması yaparken verilerinizin ne kadar büyüdüğünü görmek önemlidir. Aşağıdaki script sql backupları üzerinden databaselerinizin aylık ne kadar büyüdüğünü göstermektedir. Güzel ve kullanışlı bir script olduğu için paylaşmak istedim.

```
SELECT DATABASE_NAME,YEAR(backup_start_date) YEAR_ ,MONTH(backup_start_date) MONTH_ ,
MIN(BackupSize) FIRSTSIZE_MB,max(BackupSize) LASTSIZE_MB,max(BackupSize)-MIN(BackupSize) AS
GROW_MB,
ROUND((max(BackupSize)-MIN(BackupSize))/CONVERT(FLOAT, MIN(BackupSize))*100,2)  AS PERCENT_
 FROM
(
SELECT
        s.database_name
,       s.backup_start_date
,       COUNT(*) OVER ( PARTITION BY s.database_name ) AS SampleSize
,       CAST( ( s.backup_size / 1024 / 1024 ) AS INT ) AS BackupSize
,       CAST( ( LAG(s.backup_size )
                OVER ( PARTITION BY s.database_name ORDER BY s.backup_start_date ) / 1024 / 1024 ) AS INT
) AS Previous_Backup_Size
FROM
        msdb..backupset s
WHERE
        s.type = 'D' --full backup
        and s.database_name='ETRADE' /* DATA BASE AD */
) T GROUP BY DATABASE_NAME, MONTH(backup_start_date),YEAR(backup_start_date)
```

SQL SERVER'DA CHANGE DATA CAPTURE İLE DEĞİŞEN VE SİLİNEN KAYITLARIN LOGLANMASI

Bu makalemizde SQL Server tarafında yapılan manipülasyonların (Insert, Update, Delete) geri planda otomatik olarak kayıt altına alınmasını anlatıyor olacağız.

Bu işlerle biraz uğraşanlar için ilk akla gelen tabiki trigger yazılması. Doğru bu bir çözümdür ancak sıkıntıları vardır.
Bu sıkıntılar genel olarak şöyledir;

- Sizin yazdığınız trigger ticari programın kendisinin hata vermesine sebep olabilir ve kayıtların yapılmamasına sebep olabilir. Zira triggerlar transactionların bir parçasıdır ve trigger da gerçekleşen hata tüm transaction ı rollback yapar.

- Özellikle mevzuat değişimi gereği sıklıkla versiyon geçişi söz konusudur ve bu versiyon geçişlerinde database düzenlemesi yapıldığı için büyük ihtimal triggerlarınız silinir ve her seferinde yeniden oluşturacak scriptler oluşturmanız gerekecektir.

- Genel olarak Türkiye şartlarında dönem mali dönem bağımlı çalışmak tercih edildiği için her yılbaşında fizikken yeni tablolar oluşturulmaktadır ve bunlar için de triggerlar yeniden yazılmalıdır.

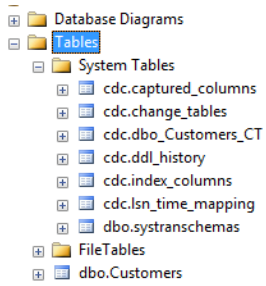
Anlaşılacağı üzere trigger meselesi etkin bir çözümdür fakat biraz zahmetlidir.
Peki, bizim yazımızın da konusu olan bu durum için bir çözüm yok mu? Birim fiyatı 5000 TL olan bir malzemenin satış faturasındaki fiyatını 50 TL olarak değiştiren bir kişiyi tespit etmenin pratik bir yolu yok mudur?
Bu noktada imdadımıza SQL Server Change Data Capture (CDC) dediğimiz özellik yetişiyor. Bu arkadaş yetenekli bir arkadaş. SQL Server'da bildiğiniz üzere tüm manipülasyon işlemleri önce Log dosyasına sonra Data dosyasına yazılır. Burada log dosyası diye bahsettiğim SQL server'ın sistem log dosyası değil database'in Log dosyasıdır (LDF).
İşte CDC sistem üzerinde Log dosyasını izler ve olan değişiklikleri hızlı bir şekilde kayıt altına alır.
Örnek olarak siz aşağıdaki gibi bir UPDATE cümlesi çalıştırdınız.
UPDATE CUSTOMERS SET ACTIVE=1
CUSTOMERS tablosunun 20 alandan oluştuğunu varsayalım oysa biz sadece bir alanı update ettik. Dolayısıyla SQL Server transaction log üzerinde sadece bir alanlık işlem hacmi söz konusu.
İşte Change Data Capture sadece bu bilgiyi okuyarak arka planda veriyi logluyor.
Siz CDC yi configure ederken belli bir süreliğine dataları loglayıp belli bir tarihten öncesini sildirebiliyorsunuz. Burada yazacağınız bir script ile önce bu datalara herhangi bir warehouse ortamına alıp daha sonra sistemden temizleyebilirsini

SQL Server Change Data Capture ile Değişiklik Yapılan Kayıtların Loglanması

1) Database imizde CDC yi enable yapıyoruz.

```
EXEC sp_cdc_enable_db
-- Tablomuzda CDC yi enable yapalım.
EXEC sys.sp_cdc_enable_table
@source_schema = N'dbo',
@source_name   = N'CS',
@role_name     = NULL,
@filegroup_name = N''
--@supports_net_changes = 1
```

2) CDC yi enable ettikten sonra system tables altında aşağıdaki tablolar oluşur.



- **cdc.captured_columns :** Adından da anlaşılacağı üzere değişikliklerin takip edileceği kritik alanları tutar. Bu tablo manuel olarak edit edilebilir durumda olup içeriği değiştirilebilir.
- **cdc.change_tables :**Hangi tabloların değişiminin takip edileceği bilgisini tutar.
- **cdc.ddl_history :**Şema bilgilerindeki değişiklikleri tutar.
- **cdc.lsn_time_mapping:** Asıl Tablo üzerinde yapılan her transaction işlemi bu tablo içerisinde tutulur ve içerisindeki lsn bilgisine göre hangi sırada yapıldığı bilgisi tutulur.

3) Kayıt ekleyelim, eklediğimiz kayıdı update edelim ve bu eklediklerimiz kayıtları silelim.

```
INSERT INTO CS (ID,NAMESURNAME) VALUES (7878,'KARABACAK GÜLTEN')
INSERT INTO CS (ID,NAMESURNAME) VALUES (7879,'KARABACAK GÜLTEN2')
UPDATE CS SET NAMESURNAME= 'KARABACAK GÜLTEN3' WHERE NAMESURNAME= 'KARABACAK GÜLTEN'
DELETE FROM CS WHERE NAMESURNAME LIKE 'KARABACAK%'
```

4)Tablolarımıza bakalım.

```
SELECT * FROM [cdc].[captured_columns]
SELECT * FROM [cdc].[ddl_history]
SELECT * FROM [cdc].[index_columns]
select * from [cdc].[lsn_time_mapping]
```

5)Loglanan kayıtlara ulaşşalım.

```
/* Log kayıtlarını ulaşmak  istediğimizde  eğer tablonun tamamına ulaşmak istiyor isek
Select burada tablo formatı cdc.<schema>_<tablename>_CT şeklinde.*/

select* from cdc.dbo_CS_CT
```

__$operation:2 Insert, 4 Update ve 1 Delete için kullanılıyor.
bu sorgu ile tüm yapılan işlemleri görebiliriz.

6)Table valued functionlar ise aşağıdaki gibi kullanılıyor.

```
DECLARE @from_lsn binary(10), @to_lsn binary(10);
--minimum lsn numarasını buluyoruz.
SET @from_lsn = sys.fn_cdc_get_min_lsn('dbo_CS');
--maximum lsn numarasını buluyoruz.
SET @to_lsn = sys.fn_cdc_get_max_lsn();
-- CDC ile ilgili işlemlerde tablo bazlı oluşan cdc function larını kullanıyoruz.
SELECT * FROM cdc.fn_cdc_get_all_changes_dbo_CS(@from_lsn, @to_lsn,'ALL');
```

Kayıt zamanını elde etmek istediğimizde, sys.fn_cdc_map_lsn_to_time functionını kullanıyoruz.

```
select sys.fn_cdc_map_lsn_to_time(__$start_lsn) as KayitZamani,* from cdc.dbo_CS_CT
```

Aşağıdaki kod 3 gün öncesine ait logları temizliyor.

```
declare @lsn binary(10);
set @lsn = sys.fn_cdc_map_time_to_lsn('largest less than or equal',getdate()-3);
exec sys.sp_cdc_cleanup_change_table @capture_instance = 'dbo_Customers', @low_water_mark=@lsn

EXECUTE AS USER ='cdc'
```

Sonuç:

- CDC gerçekten çok ihtiyaç duyulan ve çok kullanışlı bir araç.
- Sistemdeki insert, update ve delete leri loglayabiliyor.
- Eğer update cümlesinde kayıt değişmiyor ise gereksiz yer teşkil etmiyor.
- Örneğin: UPDATE CUSTOMERS SET NAME=NAME cümlesini çalıştırdığımızda herhangi bir loglama yapmıyor çünkü değişen bir şey yok.
- Sistemin çalışıyor olması için SQL Server Agent'ın mutlaka çalışması gerekir. Çünkü loğları okuyan bir job bu işleri yerine getirmektedir.
- Yazacağımız bir script ile istediğimiz tablolarda çalıştırıp istemediklerimizde çalıştırmayabiliriz. Hatta çok fazla kolon olan bir tabloda istediğimiz kolonlar için aktif hale getirirken istemediklerimizi es geçebiliriz.

SQL SERVER'DA HANGİ DB LER KULLANILIYOR? HANGİSİ KULLANILMIYOR ?

SQL Server'da birçok database iniz var. Hangisi aktif kullanılıyor, hangisi kullanılmıyor görmek istiyorsunuz. En son ne zaman erişildiğini görmek için aşağıdaki scripti görebilirsiniz.

```
CREATE TABLE #T (dbName varchar(100),last_user_seek datetime,last_user_scan datetime,last_user_lookup
datetime,last_user_update datetime)
declare @dbId as int
declare @dbname as varchar(100)
declare crs cursor for select dbid,name from sysdatabases
open crs
fetch next from crs into @dbId,@dbname
while @@FETCH_STATUS=0
begin
Insert Into #T
Select @dbname,
last_user_seek = MAX(last_user_seek),
last_user_scan = MAX(last_user_scan),
last_user_lookup = MAX(last_user_lookup),
last_user_update = MAX(last_user_update)
From
sys.dm_db_index_usage_stats
WHERE
database_id=@dbId

fetch next from crs into @dbId,@dbname
end
close crs
deallocate crs

select * from #t
drop table  #t
```

NOT IN

Verimde dışarda kalan kısımlar için kullanırım. Hangi müşteri bizimle hiç işlem yapmadı?
Son günlerde satılmayan ürünler hangileri?
Satış yapmayan personeller hangisi?

```
SELECT * FROM CUSTOMER WHERE CUSTOMER ID NOT IN (SELECT PERSONELID FROM SALES )
```

https://www.gencayyildiz.com/blog/t-sqlde-not-in-fonksiyonu-ile-tablolar-arasindaki-verisel-farki-bulma/

XML METHODLARI

XML İÇİN UYGUN VERİ METHODLARI

https:/"/www.ismailgursoy.com.tr/xml-metotlari/

ERRORLARIN KULANIMI

ERRORLA İLGİLİ TANIMLAMALARI VERİR

https://www.ismailgursoy.com.tr/try-catch-kullanimi/

KAYNAKÇA

- https://omercolakoglu.net/
- https://www.udemy.com/course/sql-ogreniyorum/
- https://www.gencayyildiz.com/blog/category/sql-server/
- https://www.youtube.com/playlist?list=PLQVXoXFVVtp2RjHt5teaBOLUcKbq2Ilbo
- https://www.youtube.com/c/SQLServerE%C4%9Fitimleri
- https://www.ismailgursoy.com.tr/category/yazilim/sql/
- https://www.udemy.com/course/sorgularla-adim-adim-sql-veri-tabani-programlama/
- https://www.youtube.com/playlist?list=PLKnjBHu2xXNP6Qa6u8GLawPnzo1brHZPP

HAZIRLAYAN: SERKAN COŞKUN