**SQL 5. Gün içeriği**

1. **Data Manipulation Language (DML) Statements**
   1. Select
   2. Insert
   3. Delete
   4. Update
   5. Truncate
2. **Commenting & Basic Where**

**Transact-SQL (T-SQL)**

SQL, düzeltilmesi veya değiştirilnesi 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.

**SQL Veri İşleme Dili (Data Manipulation Language – DML)**

Sadece tablolardaki kayıtlarla ilgili işlemler için kullanılacak ifadelerden oluşur. Yeni bir kayıt eklenmesi için INSERT ifadesini kullanırız, bir kaydın silinmesi için DELETE, güncellenmesi için UPDATE ve okunması için SELECT ifadesi olmak üzere temelde dört ifadeden oluşur.

**SELECT**: Veri seçmek

**DELETE**: Veri silmek

**UPDATE**: Veri güncellemek

**INSERT**: Veri girmek

***Eğitmen için ek bilgiler;***

**SQL Veri Tanımlama Dili (Data Definition Language – DDL)**

SQL’de verilerin tutulduğu nesneler olan tabloların yaratılmasını, silinmesini ve bazı temel özelliklerinin düzenlenmesini sağlar. En sık kullanılan DDL komutları ve kullanım amaçları aşağıdaki gibidir;

CREATE TABLE: Yeni bir tablo yaratmak

ALTER TABLE: Tabloda değişiklik yapmak

DROP TABLE: Tabloyu silmek

**SQL Veri Kontrol Dili (Data Control Language – DCL)**

SQL’de bir veri tabanı kullanıcısı veya rolü ile ilgili izinlerin düzenlenmesini sağlar. DCL komuları ve fonksiyonları şöyledir;

GRANT: Kullanıcıya yetki verir.

GRANT SELECT, INSERT ON table\_name TO user\_name;

DENY: Kullanıcı, grup veya rolü herhangi bir eylem için engeller.

DENY SELECT, INSERT ON table\_name TO user\_name;

REVOKE: Daha önce atanmış olan yetki veya engeli kaldırır.

REVOKE SELECT, INSERT ON table\_name TO user\_name;

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**SELECT Kullanımı**

SELECT deyimini SQL Server’dan veri çekmek amacı ile kullanırız. SELECT deyiminin ardından listelemek istediğimiz verilerin nerede olduğunu belirtir şekilde ifadeler kullanırız.

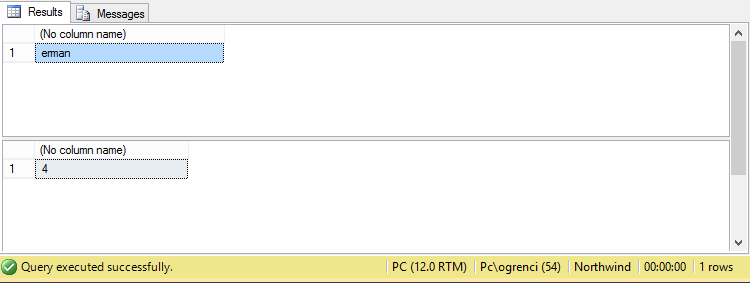
Şimdi bir SELECT deyiminin ardından rasgele yazı yazıp ve onun geriye döndürdüğü değeri görelim;

SELECT ‘Erman’

ya da

SELECT 4

Sorguyu çalıştırdğımda, bunun sonucunda geriye SQL Server, aşağıdaki gibi Erman ve 4 sonucunu döner, çünkü burada yaptığım işlem sadece SELECT sorgusunun ardından bu öğeleri seç demek. Herhangi bir veritabanı üzerinde herhangi bir tablodan veri çekmedim. Göstermek istediğim sonuç olarak veriler nasıl görünüyor bunu göstermek. Birde dikkat edilmesi gereken nokta bakın string değer için tırnak işaretleri arasında değeri belirtirken sayı için bunu direkt olarak yazdım. Eğer ‘4’ yazsa idim aslında bu da bir string türünde değer olurdu. Tabi SQL Server’da string değerin karşılığı varchar, nvarchar gibi tiplerdir.



Bir tablo üzerinden veri çekeceğimiz zaman aşağıda ki kalıbı kullanırız;

SELECT *sütun\_adı, sütun\_adı*  
 FROM *tablo\_adı*

// Bir tabloda yer alan sütun isimleri bilinmeden de tüm sütunların seçilmesi için **\*** ifadesi kullanılabilir.

Bazı durumlarda, tabloda yer alan bütün sütunları görmek yerine sadece bazı sütunları görmemiz gerekiyor olabiliyor. Bu duruma *izdüşürme (projection)* denir. Sadece bazı sütunlarda yer alan değerleri görmek için SELECT’ten sonra bu sütunların adlarını araya virgül koyarak yazmak yeterlidir.

SELECT FirstName,LastName

FROM Employees

***Eğitmen için ek bilgiler;***

Birden fazla tekrarlayan kayıt döndüren SELECT işlemlerinde, her bir kaydın tekil olarak yer almasını istiyorsak DISTINCT deyimi kullanılır.

SELECT DISTINCT Country

FROM Employees

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**INSERT Kullanımı**

Bir tabloya kayıt eklemek için kullanılır. En basit genel kullanım şekli;

INSERT INTO *tablo\_adi (sutun1,sutun2)*

VALUES *(deger1,deger2)*

veya

Aşağıdaki yöntemde tabloda bulunan bütün sütunlar(null geçilemeyen) için değer girmemiz gereklidir.

INSERT INTO *tablo\_adi*

VALUES *(deger1,deger2, ...)*

Örnek olarak;

INSERT INTO Employees (FirstName,LastName)

VALUES (‘Erman’,’Zöhre’)

**UPDATE Kullanımı**

Update ifadesi ile satır ya da satırların değerleri değiştirilir. Bir şart verilmezse, değişikliği bütün satırlar için yapar. Veri kaybı ile karşılaşmamak için şart olup olmadığı gözden kaçırılmamalıdır. Genel olarak kullanımı şu şekildedir.

UPDATE *tablo\_ismi*

SET *alan=deger*

WHERE *şart\_tanımları*

Örnek;

UPDATE Employees

SET FirstName=’Erman’

WHERE FirstName=’Andrew’

**DELETE Kullanımı**

Kayıt silmek için DELETE ifadesi kullanılır. DELETE için bir şart verilmezse bütün kayıtların silineceği akıldan çıkarılmamalıdır. Genel kullanımı şu şekildedir;

DELETE FROM *tablo\_ismi*

WHERE *şart\_tanımları*

\*\* *DELETE ifadesi çalıştırılmadan önce WHERE cümleciğinin amacınıza uygun olup olmadığını mutlaka kontrol edin. Düzgün planlanmamış bir DELETE ifadesi, plan dışı kayıtların silinmesi ile meydana gelebilecek veri kayıplarına neden olabilir.*

Örnek;

DELETE FROM Employees

WHERE FirstName=’Nancy’

**TRUNCATE Kullanımı**

Bir tabloda yer alan tüm kayıtları silmek ama bu silme işlemleri için de satır bazlı birer *transaction log* oluşturulmasının istenmediği durumlar olabilir. Bu tür durumlarda **TRUNCATE TABLE** komutu kullanılır.

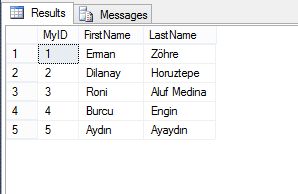
Genel kullanımı şu şekildedir:

TRUNCATE TABLE *tablo\_ismi*

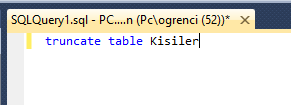
*\*\*Burada ki* ***tablo\_ismi****, içeriğini satır satır loglamaksızın boşaltacağımız tablo adını ifade eder.*

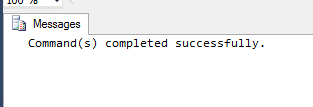
Örnek: Elimizde kişiler tablomuz olsun.

****

****

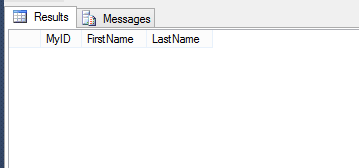
Bu tabloyu TRUNCATE TABLE ile uçuralım.

****

****

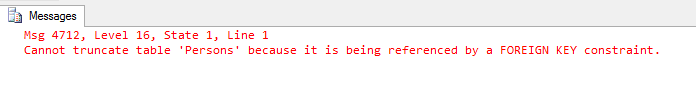
Tekrar SELECT ile tablomuza baktığımızda içinde ki bütün bilgilerin uçtuğunu görmüş oluruz.

****

****

**TRUNCATE ve DELETE FARKLARI**

* TRUNCATE TABLE ile WHERE cümleciği olmayan DELETE komutu aynı işlevi yapar; bir tablonun tüm satırlarını boşaltır.
* TRUNCATE TABLE ile boşaltılan bir tablonun IDENTITY değeri sıfırlanır ve yeniden başlangıç değerinden artmaya başlar. DELETE ifadesinde ise kaldığı yerden devam eder.
* Bir tabloyu gösteren *foreign key constraint* varsa, bu kısıtlayıcı olduğu sürece TRUNCATE TABLE deyimi bu tablo üstünde çalıştırılamaz.

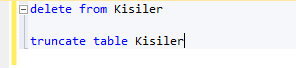


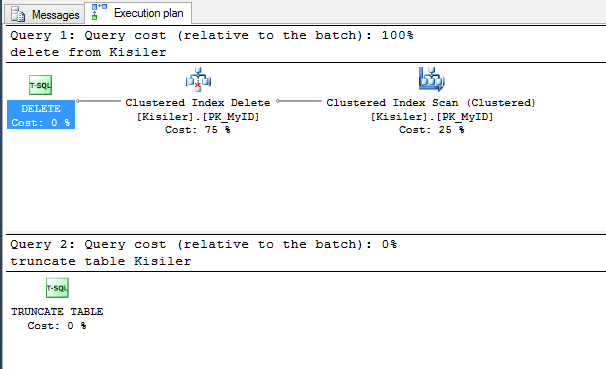
* TRUNCATE ile silinen tabloda DELETE TRIGGER’ı varsa devreye girmez.
* TRUNCATE komutu ile parçalı silme yapılamazken DELETE komutu ile WHERE ifadesi kullanılarak parçalı silme yapılabilmektedir.

DELETE komutu ile yapılan kayıt silme işleminde her silinen kayıt Transaction Log da loglanmaktadır (Bu yüzden log dosyamız hızlı şişer). TRUNCATE de ise loglama işlemi kayıt bazında yapılmamaktadır.

Bu yüzden TRUNCATE komutu DELETE komutuna oranla büyük tablolarda inanılmaz hızlıdır.

Son olarak DELETE ve TRUNCATE in execution planlarını karşılaştıralım;





**COMMENTING (AÇIKLAMA -YORUM SATIRLARI)**

**MsSQL**’de açıklamaları 2 şekilde yaparız.

1 - Satır bazlı açıklama için -- (iki yanyana tire) kullanılır.

2 - Parağraf olarak açıklama yapmak için /\*…\*/ kullanılır. Bu alan arasına yazılan şeyler server tarafından değerlendirilmeye alınmaz.

**TEMEL WHERE KULLANIMI**

**WHERE** kelimesi kullanılarak, hangi satırların geleceği, hangi satırların gelmeyeceği tanımlanabilir. Genel kullanımı şu şekildedir:

SELECT sütun\_isimleri

FROM tablo\_ismi

WHERE şart\_tanımları

Aşağıdaki örneklerde WHERE kelimesi kullanımına örneklerdir;

1 - Basit bir eşitlik kullanarak satırı getirme;

SELECT ProductID, ProductName

FROM Products

WHERE ProductName = 'chai'

*Kullanılabilecek diğer operatörler;*

< > Eşit değildir. (Bazı SQL versiyonlarında != )

> Büyüktür

< Küçüktür

>= Büyüktür veya eşittir

<= Küçüktür veya eşittir

2- Bir metnin içinden bir kelimeyi içeren satırı bulma

SELECT ProductID, ProductName

FROM Products

WHERE ProductName LIKE '%cajun%'

-- % işareti başa gelirse kelimeden öncesini yoksay, sonuna gelirse kelimeden sonrasını yoksay anlamındadır. (like ile kullanılır % işareti)

3- Bir metnin içinden bir kelimeyi içermeyen satırı bulma

*LIKE deyiminin tersidir aslında. Bir durumun benzememe halini kontrol eder.*

SELECT EmployeeID, FirstName, LastName

FROM Employees

WHERE FirstName NOT LIKE '%an%'

4 – Karşılaştırma operatörü kullanılarak satırı getirme

SELECT ProductID, ProductName

FROM Products

WHERE ProductID < 12

5 – Üçlü koşullardan herhangi birini karşılayan satırı bulma

SELECT ProductID, ProductName

FROM Products

WHERE ProductID = 2

OR ProductID = 4

OR ProductName like 'louisiana%'

6 – Çoklu koşulların hepsini karşılayan satırı bulma

SELECT ProductID, ProductName, UnitPrice

FROM Products

WHERE ProductID >20

AND ProductName LIKE ('G%')

AND UnitPrice < 50

7 – Listedeki değerleri içeren satırları bulma

SELECT ProductID, ProductName, UnitPrice

FROM Products

WHERE ProductName IN ('Chai', 'Konbu', 'Chang')

\*\* Yukarıdaki sorgunu tersi olarak, NOT IN diyerek belirtiğimiz ürünler dışındaki ürünleri elde edebiliriz.

8 – İki değer arasındaki koşulu içeren satırı bulma

SELECT OrderID, OrderDate, ShipCountry

FROM Orders

WHERE OrderDate BETWEEN '1996-07-04' AND '1996-07-09'