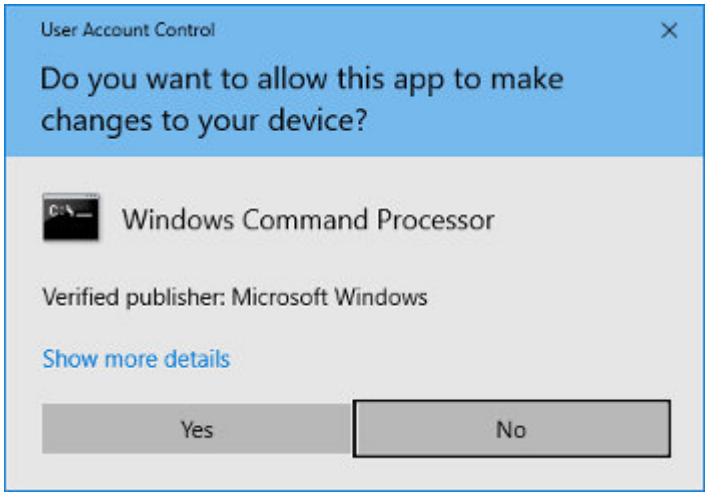
Uzak Bilgisayar Yerel Gruplardan Kullanıcı Silme

Sistem yönetiminde istemci bilgisayarlardaki özellikle “Yöneticiler” grubunda yer alan kullanıcıların kontrol edilmesi son derece önemlidir. Bu kontrol sonucunda da gerekli durumlarda bu kullanıcıların silinmesi gerekiyor. Bu işlem için bir yazılım geliştirilmesi gerekiyorsa gerekli olan C# metotu aşağıdaki gibidir.

```
public bool RemoveUserFromAdminGroup(string
computerNameVeyaIp, string silinecekKullanıcı)

        {
            try

            {

                var de = new DirectoryEntry("WinNT://" +
computerName);

                var objGroup =
de.Children.Find("Administrators", "Group");
//Administrator: Kullanıcısı silinecek grup
//Group: Statik bir değerdir. Administrator öğesinin grup
olduğunu belirtiyor.

                foreach (object member in
(IEnumerable)objGroup.Invoke("Members"))

                {

                    using (var memberEntry = new
DirectoryEntry(member))

                        if (memberEntry.Name == user)
```

```
                                objGroup.Invoke("Remove", new[] {
memberEntry.Path });

                }



                objGroup.CommitChanges();

                objGroup.Dispose();



                return true;

            }

            catch (Exception ex)

            {

                MessageBox.Show(ex.ToString());

                return false;

            }

        }
```

Yönetici İznine (UAC) Takılan Program Sorunu



Çalışırken bazı programlar görünürde yönetici iznini gerektirecek her hangi bir işlem yapmamalarına rağmen yönetici izni istemektedirler. Özellikle bilgi güvenliği noktasında prosedürler yürüten firmalarda da kullanıcı bilgisayarı üzerinde yönetici olmadığından uygulamayı çalıştırmakta başarısız olmaktadır.

Programın yönetici izni istemesinin temelinde yatan neden, çalışırken ya sistem ayarlarına müdahale ediyor olması yada sistem dosyalarında işlem yapmaya çalışmasıdır. Burada sistem dosyaları sözü aklı karıştırmasın. Program “Program Files” varsayılan klasörüne yüklendiğinde kendi dosyaları da “Program Files” klasörünün özelliği gereği Windows tarafından sistem dosyası olarak işaretlenmektedir ve bu dosyalar üzerinde işlem yapabilmek için yönetici izni gerekmektedir.

Çözüm olarak karşımıza iki seçenek çıkmaktadır;

- Kullanıcıya yerel yönetici haklarını vermek ki bu özellikle bilgisayar noktasında tecrübesi olmayan kullanıcılar tarafında ayrı bir güvenlik açığına sebep olmaktadır.
- Diğer bir seçeneğimiz de programın kurulum yolunu

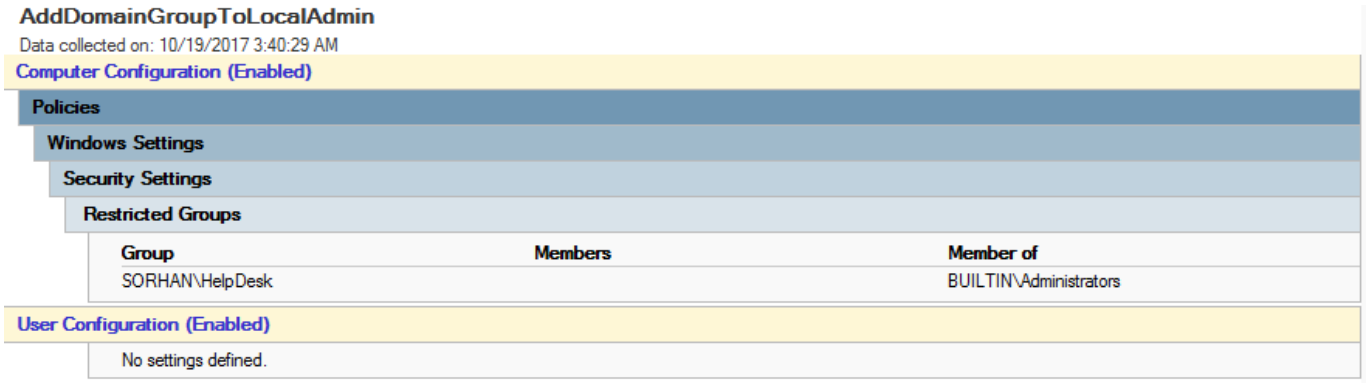
değiştirmektir. Örneğin C:\ diski altında oluşturacağımız bir klasörü kurulum yolu olarak verirsek, oluşturduğumuz bu klasör sistem klasörü olmadığından uygulamamız da sıkıntısız çalışacaktır.

Domain Yapısındaki Grupları Client Local Gruplara Ekleme

Domain yapılarında yapılan işlemlerden biri de kullanıcılardan local adminlik dediğimiz yönetici haklarının alınmasıdır. Bu işlemi yapmada ki amaç, kullanıcının bilgisayarı amacı dışında kullanmasına engel olmaktır. Bu işlemi yaparken de, özellikle bilgi işlem departmanının local admine atanması gerekmektedir. Bu atamayı yapmanın çeşitli yolları olmaktadır;

- Bilgi teknolojileri departmanının hepsini, Domain Admins grubuna dahil etmek: Bu seçenek güvenlik tehlikesini client bazından çıkarıp direk bütün yapı seviyesine çıkarır. Departmanda yer alan özellikle stajyer ve yeteri bilgiye sahip olmayan yeni çalışanların oluşturacağı güvenlik tehdidi göz ardı edilemez. Bu seçenek kabulümüz değil dolayısıyla
- Her kurulan yeni client bilgisayara, kurulumdan sonra BT departmanındaki bütün kişileri elle yöneticiler grubuna eklemek. Bu yönteminde dezavantajları: Kişileri tek tek eklerken kişiler unutulabilir, departman değiştiren kişiler bilgisayarlarda yönetici olarak kalmaya devam edecek…
- BT departmanı için domain yapısında bir grup oluşturup

departman kişileri bu gruba üye edilir. Bu işlemden sonra Group Policy üzerinden oluşturulan bu grup clientlara otomatik olarak yönetici grubuna eklenir. Bu işlem için gereken group policy kuralı aşağıdaki gbidir.



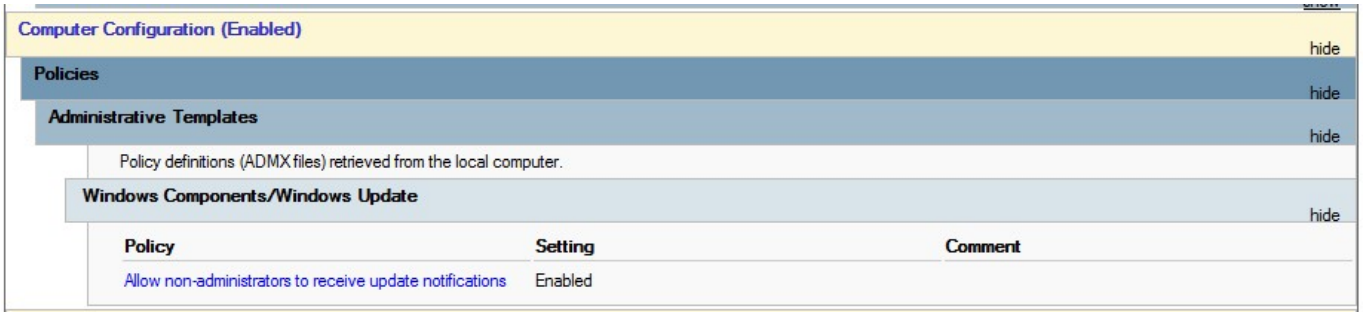
Kuralın sol tarafında yer alan “Group” başlığı clientlara eklenecek olan domain üzerinde tanımlı olan gruptur. SORHAN burada domain adını temsil etmektedir. HelpDesk de BT departmanı grubudur. Sağ tarafta yer alan “Member of” başlığı da grubun client tarafında nereye ekleneceğini ifade eder. BUILTIN ifadesi client makineyi ifade eder ve sabittir. Administrators ifadesi de clent makine üzerinde tanımlı Administrators grubunu ifade eder.

GP Üzerinden Herkese Güncelleme Yetkisi Verme

Bilişim sistemlerinin en önemli konularının başında sistemin güvenliğidir. Konu güvenlik olunca da makinelerde yetkilendirme devreye giriyor. Her şirkette olması gereken yetki kısıtlarından biri de bilgisayarlarda local adminliklerin olmaması kuralıdır. Kullanıcılardan local admin yetkileri alındıktan sonra ortaya bir sıkıntı daha çıkmaktadır: o da güncellemeleri yüklemek için kullanıcılardan yönetici izni istemek…

Güncelleme yüklenmeyince sistem açıklarını kapatan güncellemeler sisteme yüklenmemektedir.

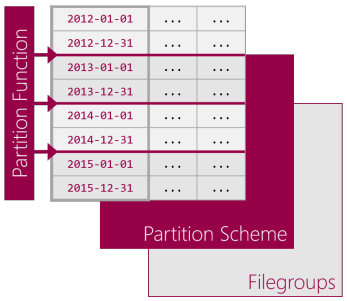
Çözümümüz şu olacak: Group policy üzerinden bütün kullanıcılara güncelleme yükleme izni vermemiz gerekmektedir. İlgili GP ve gerekli değeri aşağıdaki resimdeki gibidir.



Visual Studio İle GitHub Bağlantısı

Windows Server 2012 Üzerine FileZilla FTP Server Kurulumu (Video)

SQL Server Büyük Tabloları Bölmek



Bir güzel SQL Server özelliği ile daha beraberiz. Veri tabanları işlevleri gereği milyonlarca hatta milyarlarca kayıt tutabiliyorlar. Bu kayıtlardan bazıları üzerinden zaman geçtikçe sorgulanmayan veriler olmaya başlıyor. Örneğin online alışveriş sitelerinde üç dört yıl önceki siparişler zorunlu bir işlem olmadığı sürece sorgulanmamaya başlar. Ama sorgularda bu kayıtlar

da sorgulandığından sorgular yavaş çalışmaya başlar. Bu durumlar da SQL Server Partitioned Table özelliği ile büyük tabloları bir özelliğine göre farklı dosyalara ayırabiliriz. Böylece gelen sorguda bütün verileri değil de sadece ilgili verinin olduğu dosya üzerinde çalışılır. Saatlerce sürebilecek sorgular bu sayede saniyeler içerisinde bite bilmektedir.

Bölünmüş Tablo Oluşturma

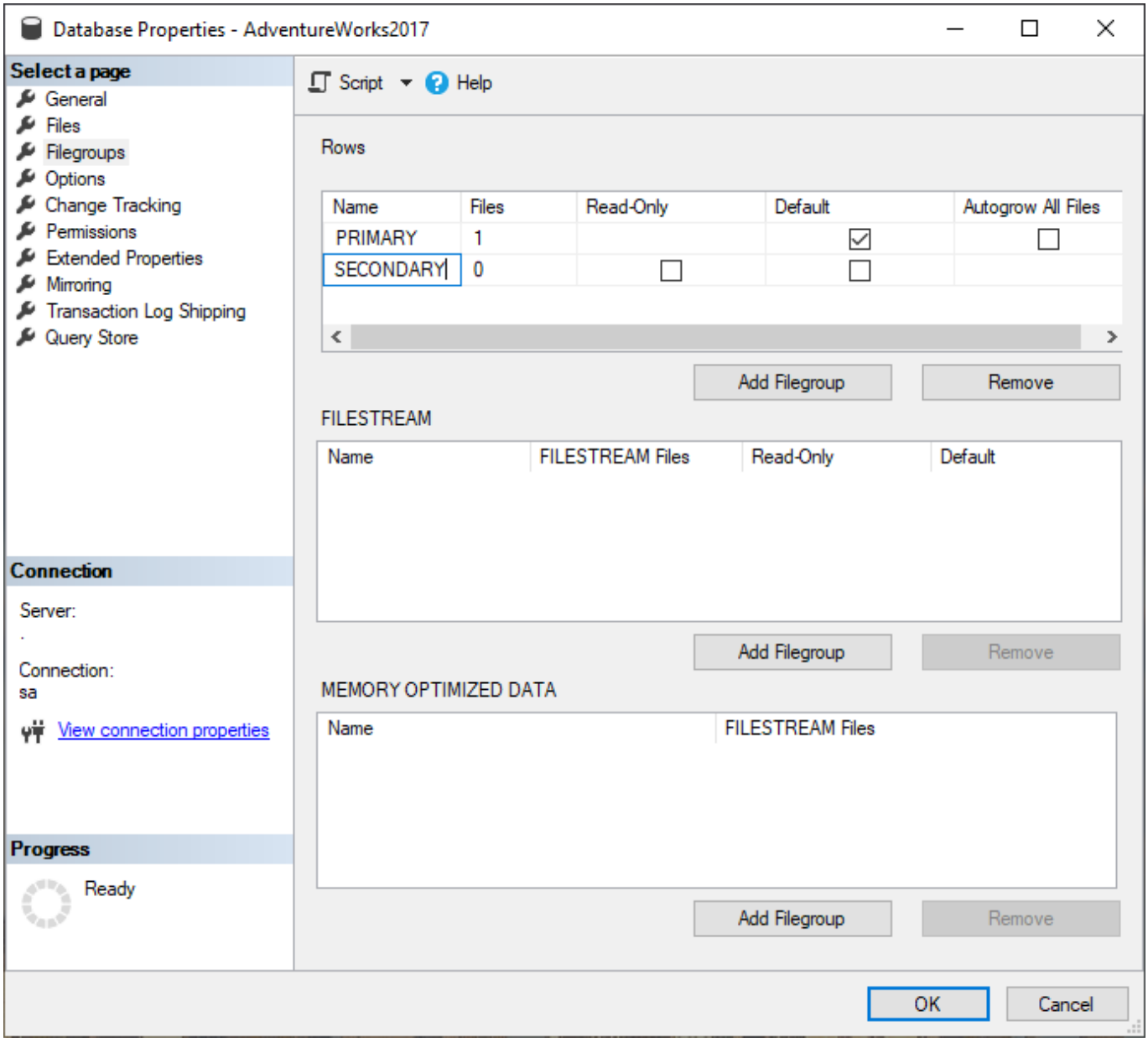
SQL Server Management Studio veya T-SQL ile tablo bölünebilir.

Tablo bölme işlemi genel olarak dört adımda olmaktadır.

- Bölünecek tablonun verilerini tutacak FILEGROUPS ve dosyaların oluşturulması
- Bölünme kurallarını oluşturacak fonksiyonun oluşturulması
- Bölünmüş dosyaların tutulacağı schemanın oluşturulması
- Tablonun oluşturulan fonksiyon ile schema üzerinde bölünmesi

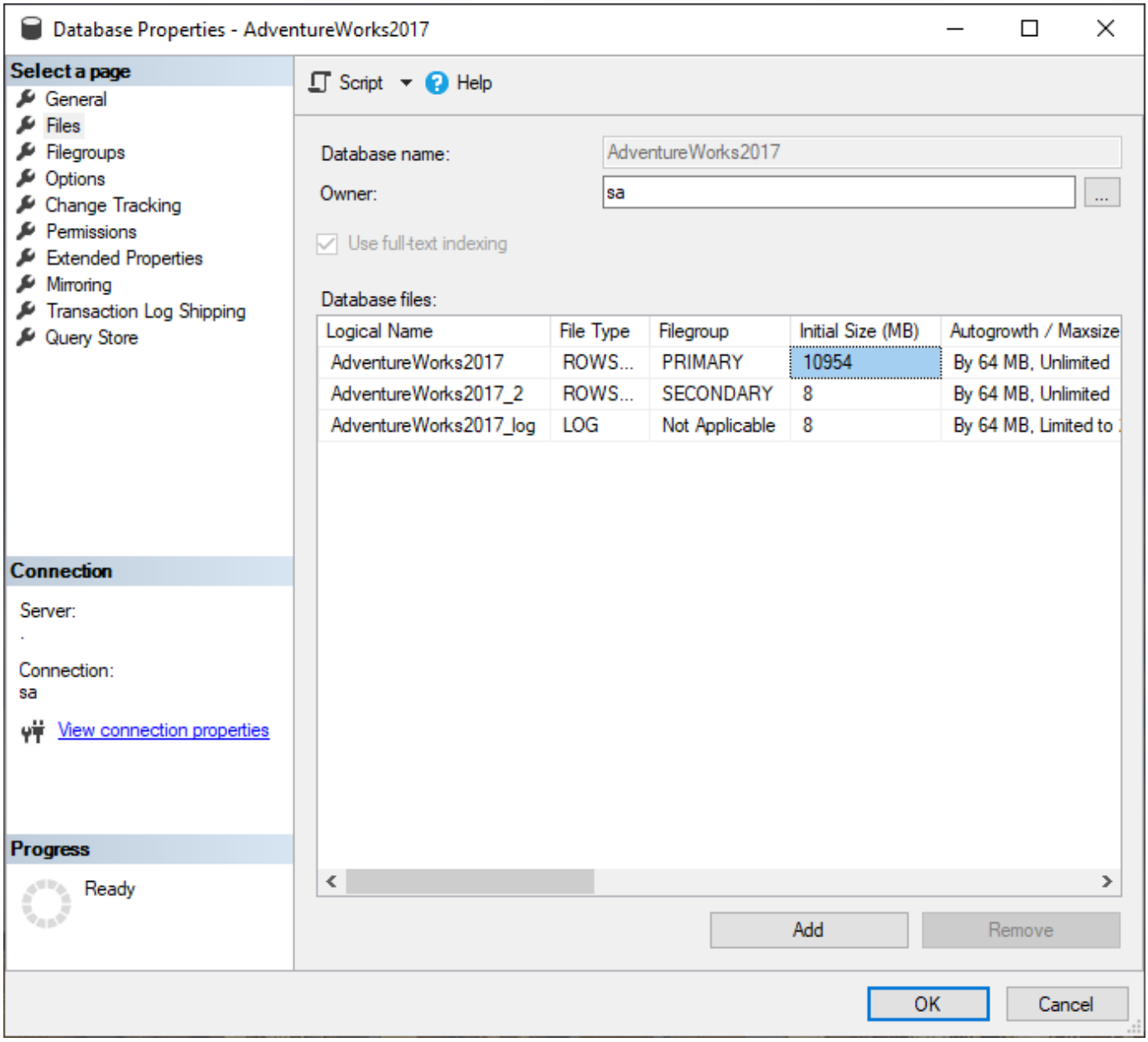
SQL Server Management Studio İle Bölme İşlemi

İlk işlem olarak ilgili veri tabanını sağ tıklayıp özellikler (Properties) ekranını açarak aşağıdaki ekran görüntülerine göre FILEGROUP ve File ekliyoruz.



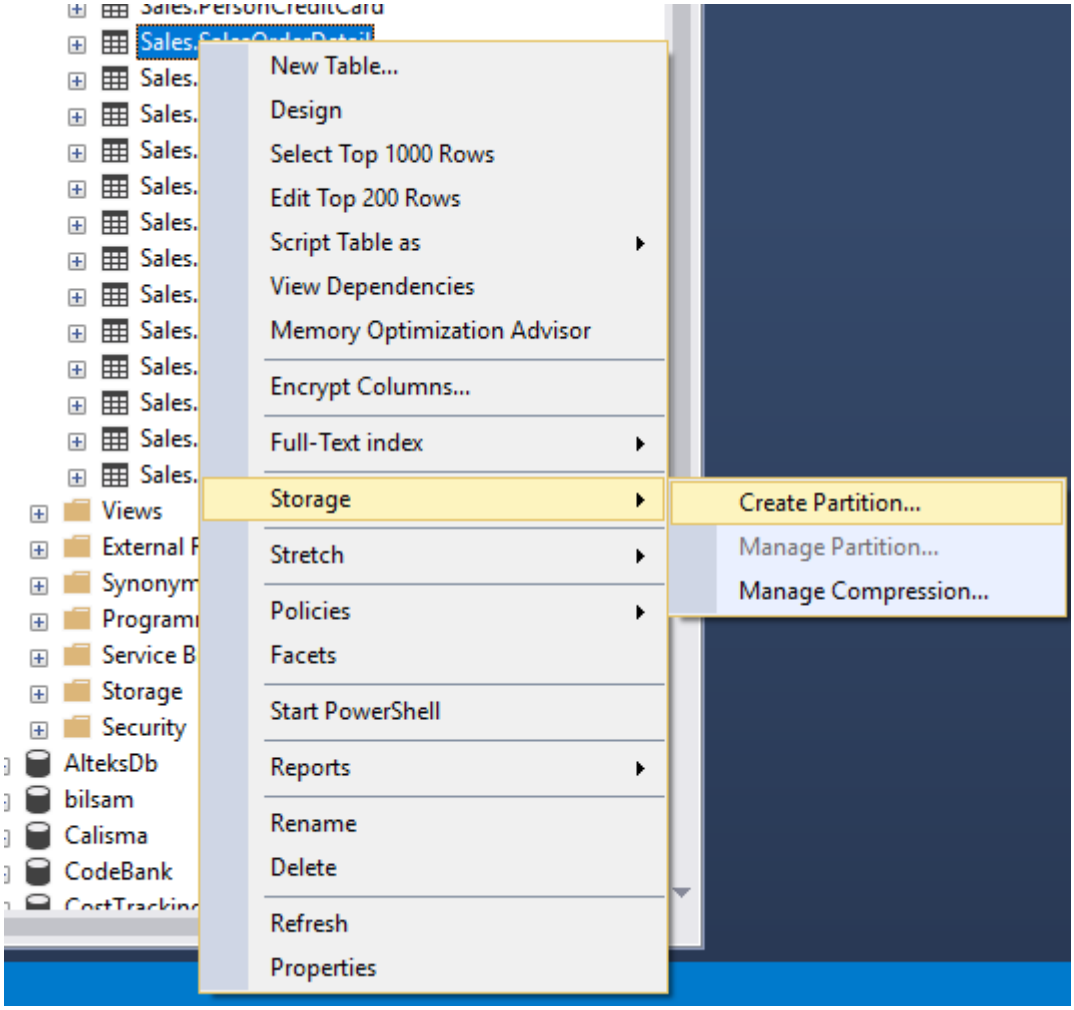
FILEGROUP Ekleme

“Rows” kısmında “Add Filegroup” butonuna tıklayarak bir filegroup ekliyoruz. Filegroup ekledikten sonra bu filegroup içerisinde yer alacak file ekliyoruz.



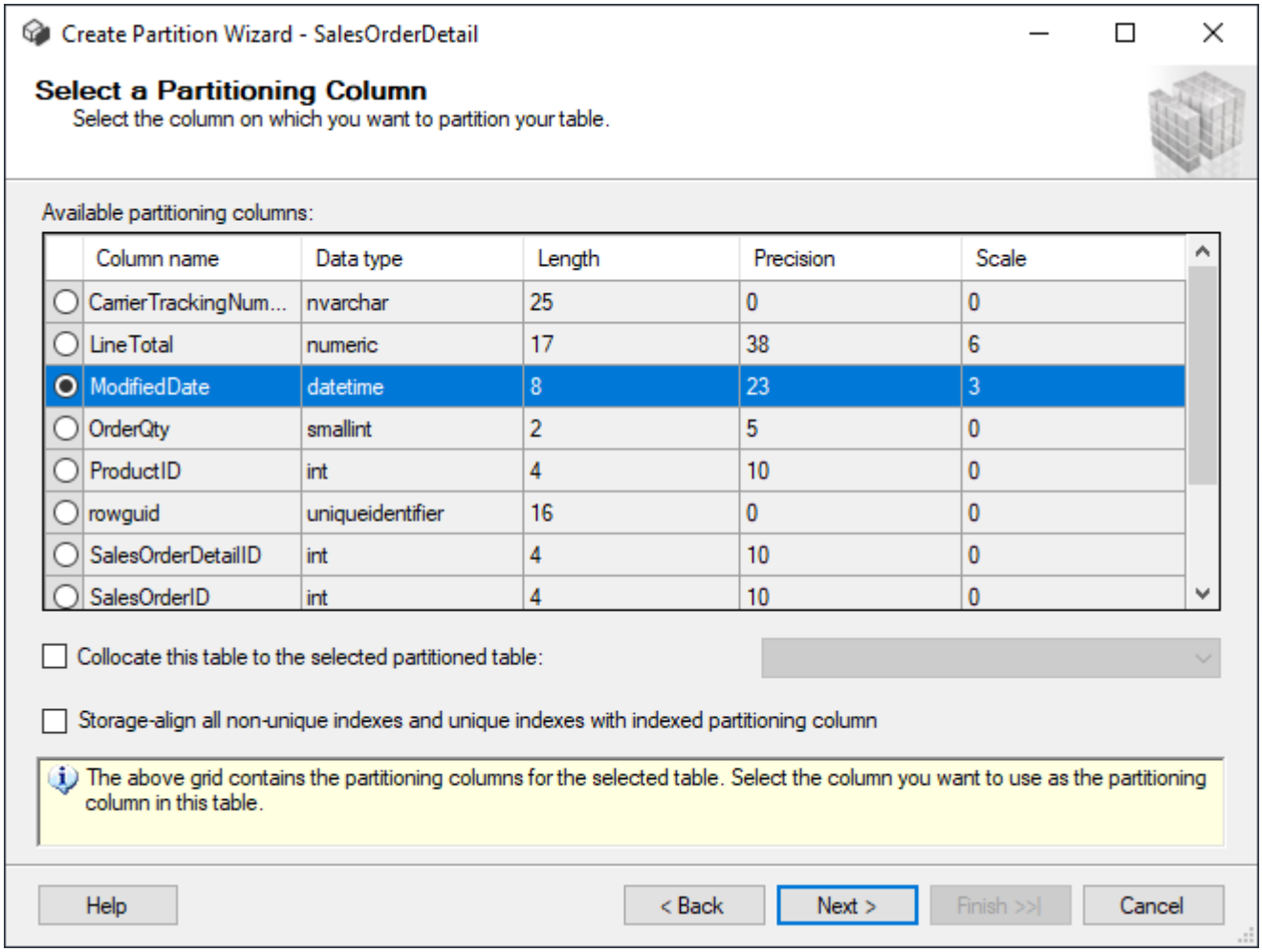
FILE Ekleme

“Add” butonu ile bir önceki adımda oluşturduğumuz filegroup içerisinde yer alacak bir dosya ekliyoruz. Burada dikkat etmemiz gereken nokta dosyanın ndf uzantılı olması gerektiğidir.



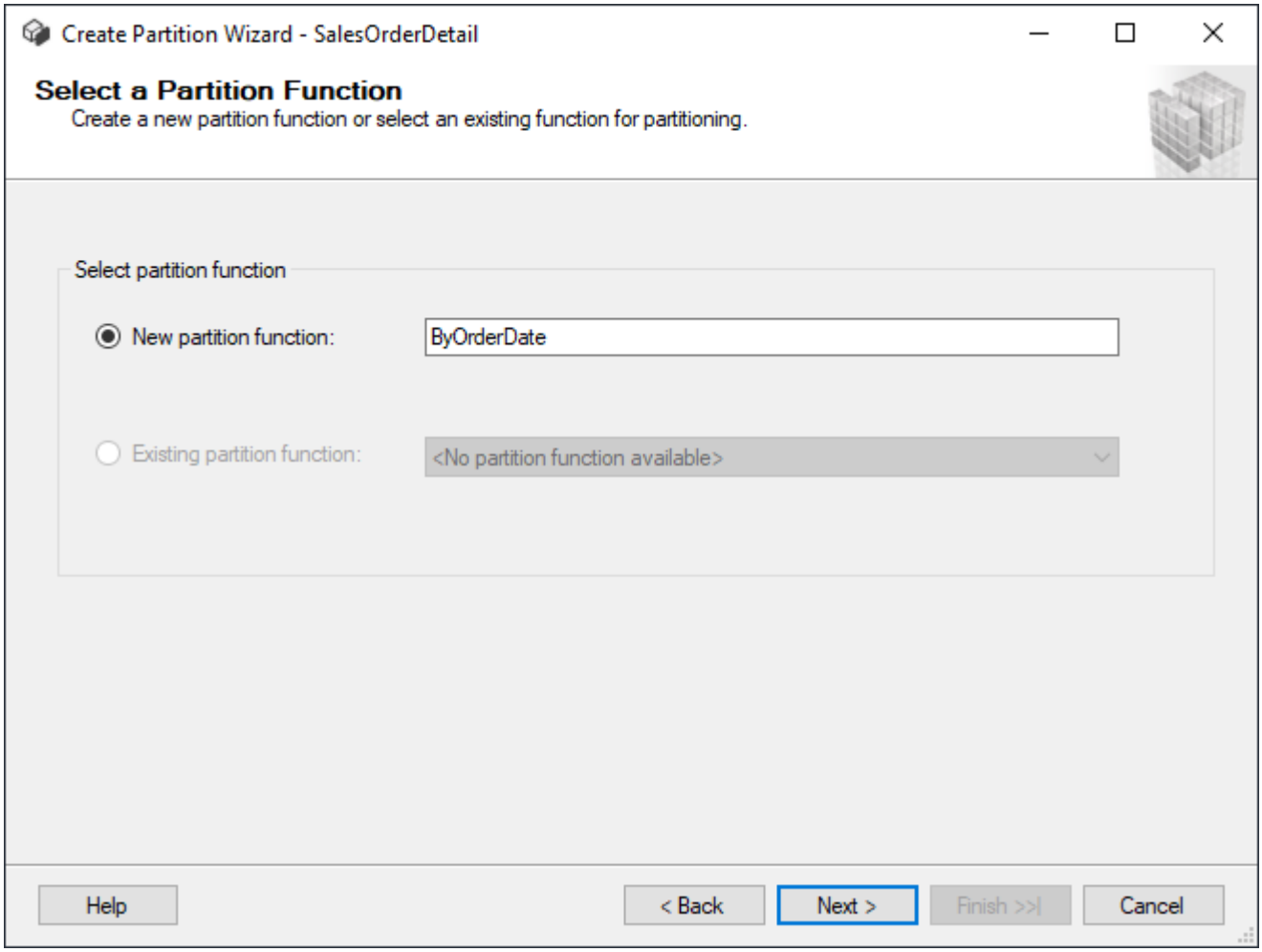
Tablo Bölme İşlemi Başlatma

Filegroup ve file oluşturma işlemlerinden sonra tabloyu bölme işlemine başlıyoruz. Bölünecek tablo üzerine sağ tıklayarak Storage > Create Partition… yolunu izleyerek işleme başlıyoruz.



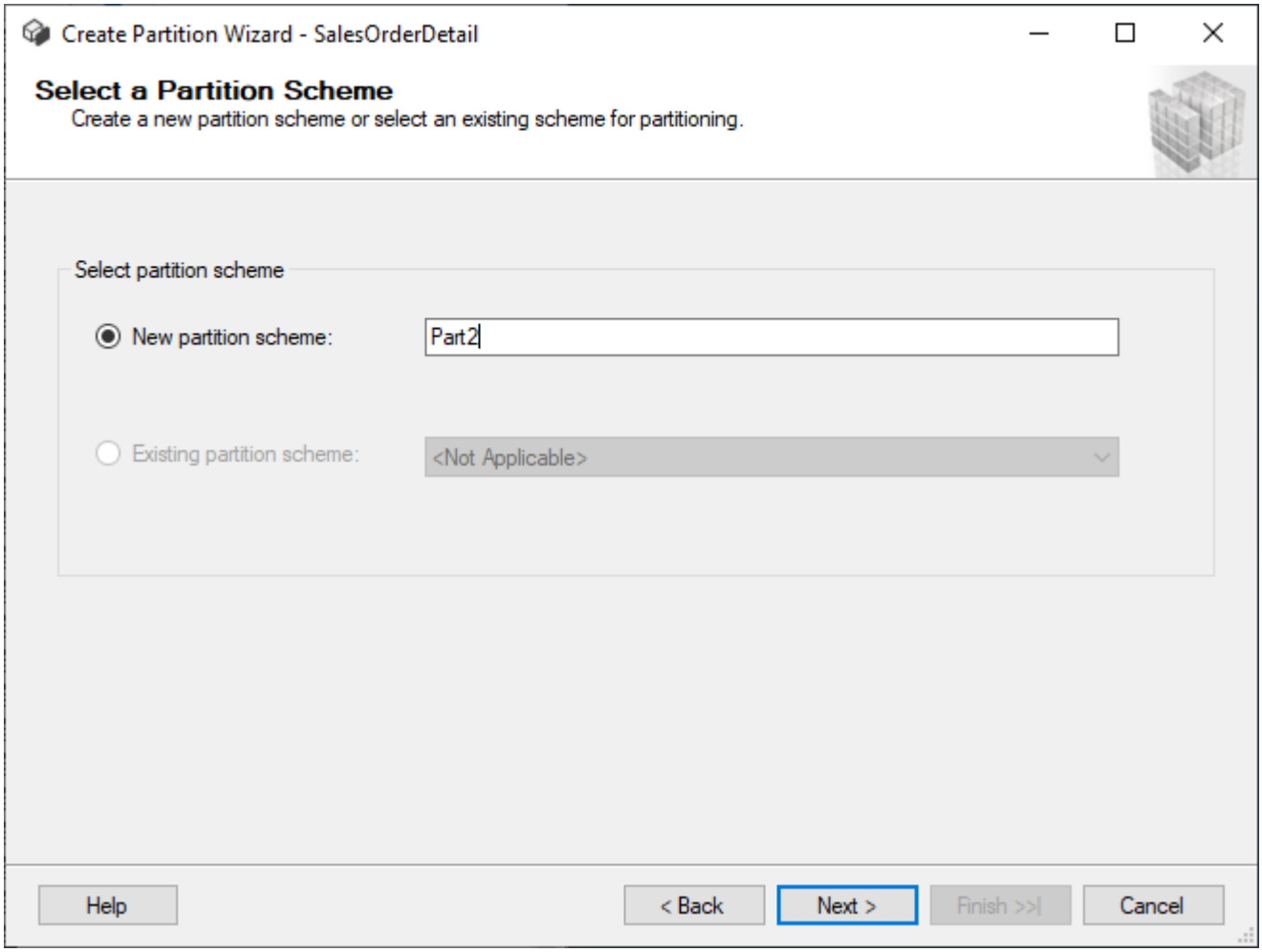
Üzerinde Bölme Yapılacak Kolon Seçimi

Açılan “Select a Partitioning Column” ekranında üzerinde bölme şartının çalışacağı kolonu seçiyoruz. Mantıksal olarak gruplanabilen her hangi bir kolon seçilebilir.



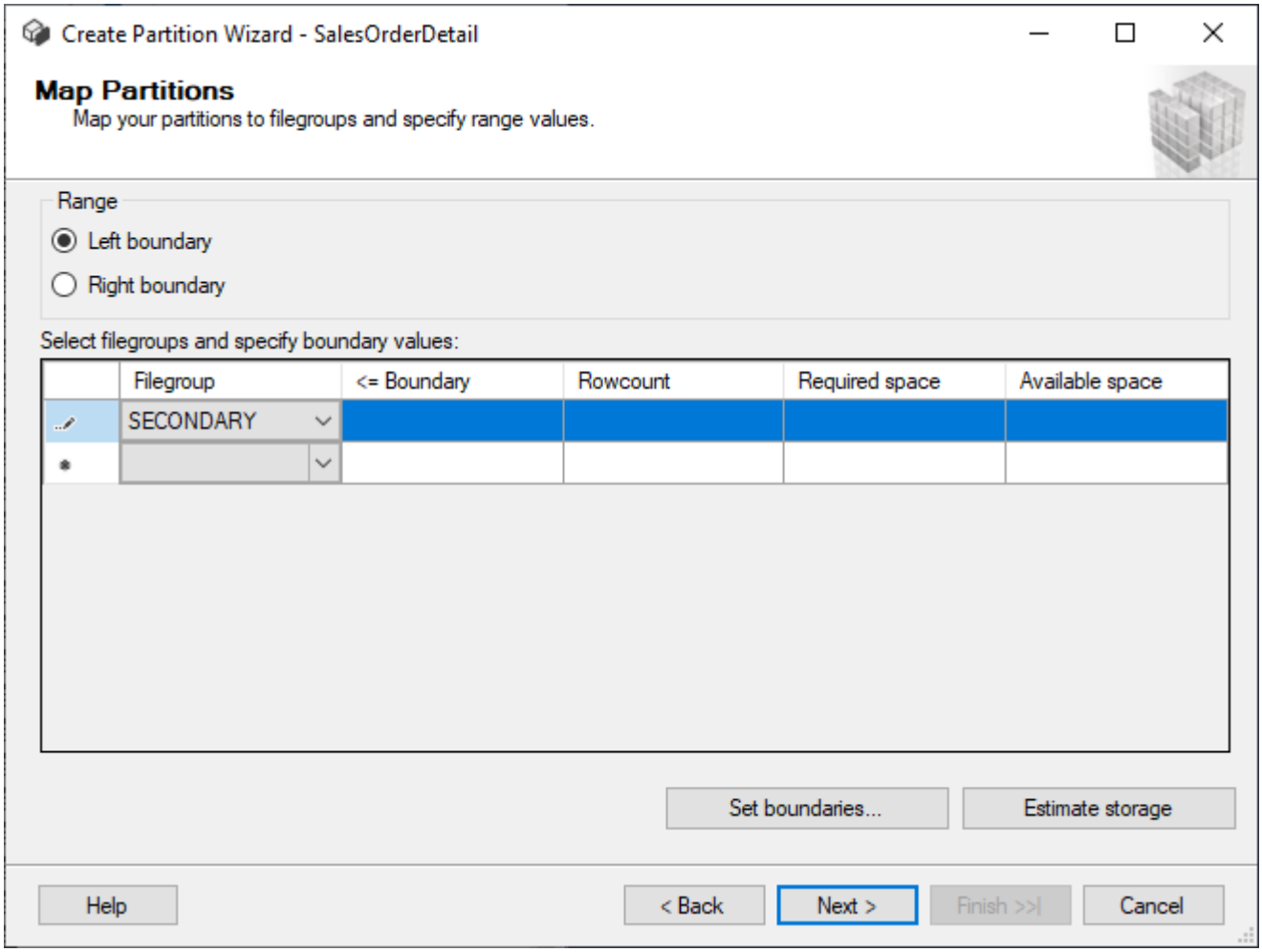
Partition Function

“Select a Partition Function” ekranında bölme kuralını içeren partition fonksiyonunu seçiyoruz. Daha önceden oluşturulan bir fonksiyon var ise “Existing partition function” seçeneği ile seçiyoruz. Oluşturulan fonksiyon yok ise “New partition function” seçeneği ile fonksiyon adını giriyoruz. Sistem girilen isimde fonksiyonu otomatik oluşturacaktır.



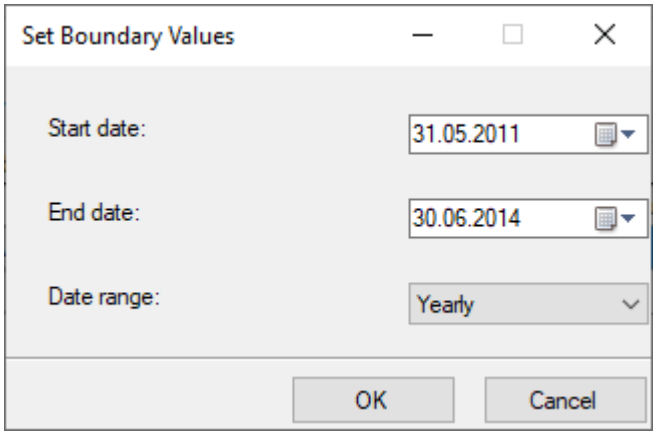
Şema Seçimi

“Select a Partition Scheme” ekranında da bölünme işlemini tutacak schema seçimi yapıyoruz.



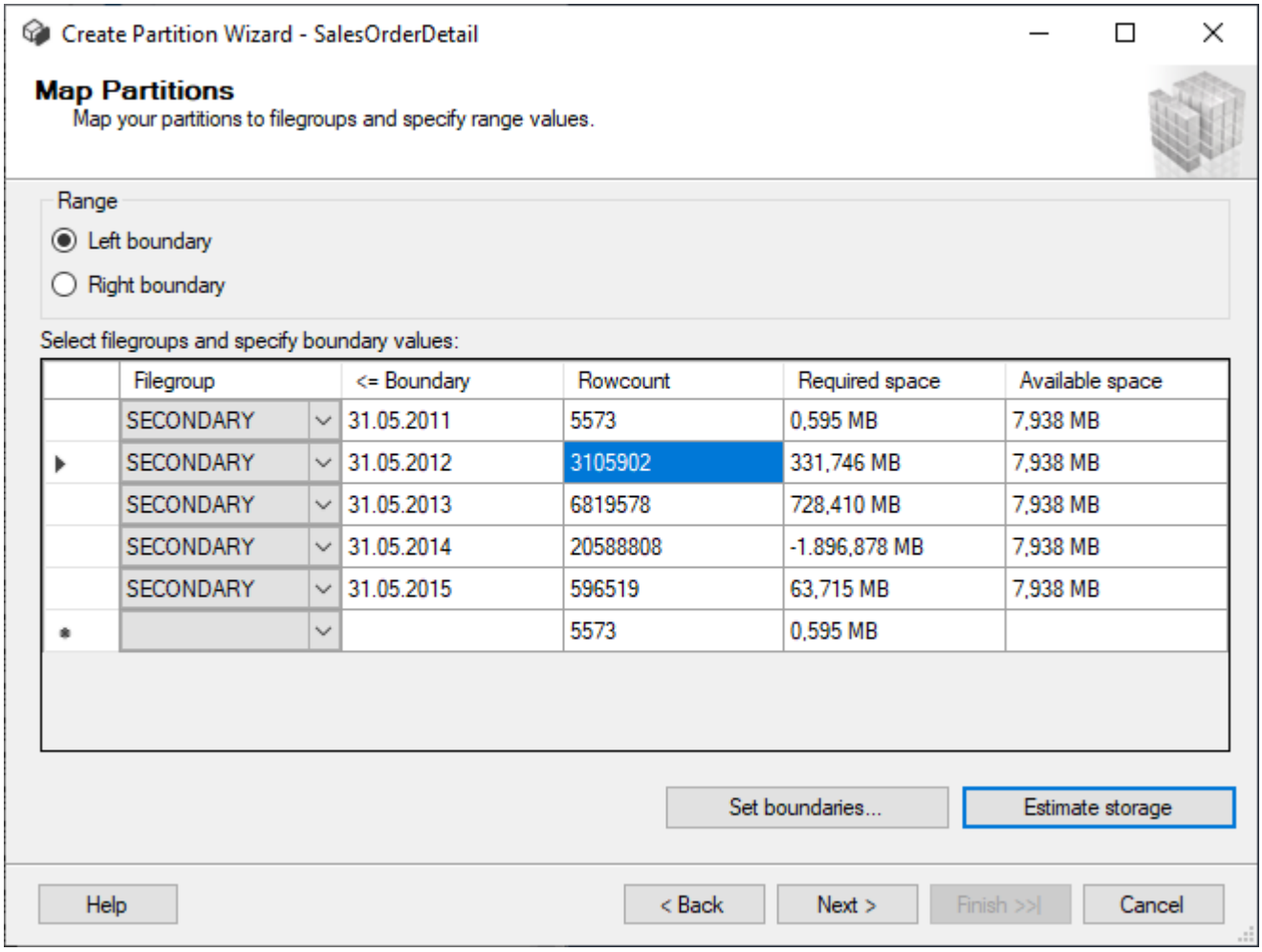
Bölme Kurallarının Belirlenmesi

“Map Partition” ekranında verilerin hangi kurala göre hangi dosyalara bölüneceği kurallarını belirliyoruz. Tarih verisi içeren kolona göre bölme işlemi yapıyor isek “Set Boundaries…” butonuna tıklayarak aralıkları otomatik hesaplatabileceğimiz bir diyalog penceresi açabiliriz.



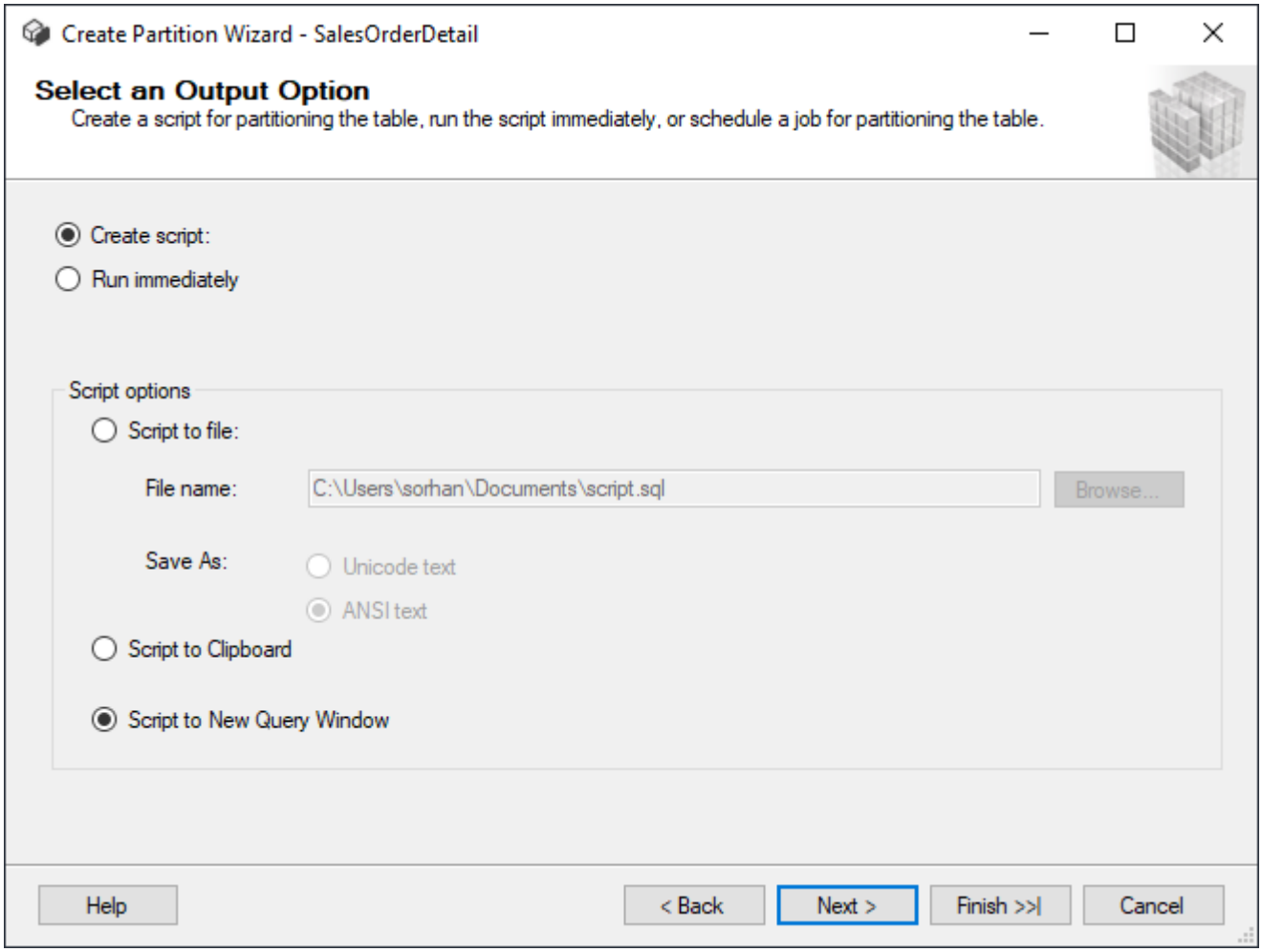
Set Boundaries…

“Set Boundaries…” ekranında başlangıç ve bitiş tarihlerini girdikten soran bölünmenin aralığını seçiyoruz. Örneğimizde verileri yılına göre böleceğimizi seçmişiz mesela.



Set Boundaries… işlemi sonrası

“Estimate storage” butonuna tıklayarak verilerin bölünme sonrası durumlarını ön izleyebilirz.



İşlem Başlatma

Son adım olarak işlemin ne zaman yapılacağı ile ilgili seçimi de yaptıktan sonra tablo bölme işlemi tamamlanmış olacaktır.

T-SQL İle Tablo Bölme

```
USE [AdventureWorks2017]
GO
BEGIN TRANSACTION
CREATE PARTITION FUNCTION [ByOrderDate](datetime) AS RANGE
LEFT FOR VALUES (N'2011-05-31T00:00:00',
N'2012-05-31T00:00:00', N'2013-05-31T00:00:00',
N'2014-05-31T00:00:00', N'2015-05-31T00:00:00')
```

```
CREATE PARTITION SCHEME [Part2] AS PARTITION [ByOrderDate] TO
([SECONDARY], [SECONDARY], [SECONDARY], [SECONDARY],
[SECONDARY], [SECONDARY])

ALTER TABLE [Sales].[SalesOrderDetail] DROP CONSTRAINT
[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] WITH (
ONLINE = OFF )


ALTER TABLE [Sales].[SalesOrderDetail] ADD  CONSTRAINT
[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID] PRIMARY
KEY NONCLUSTERED
(
        [SalesOrderID] ASC,
        [SalesOrderDetailID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, ONLINE = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]


CREATE CLUSTERED INDEX
[ClusteredIndex_on_Part2_636815093769025611] ON
[Sales].[SalesOrderDetail]
(
        [ModifiedDate]
)WITH (SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF, ONLINE =
OFF) ON [Part2]([ModifiedDate])


DROP INDEX [ClusteredIndex_on_Part2_636815093769025611] ON
[Sales].[SalesOrderDetail]

COMMIT TRANSACTION
```

SQL Server Veri Girme (Insert)

SQL Server üzerinde tanımlı tablolara veri girişi INSERT komutu ile olmaktadır. INSERT komutunun kullanım şekli aşağıdaki gibidir:

```
INSERT INTO tablo_adı (kolon1, kolon2, kolon3, ...)
VALUES (değer1, değer2, değer3, ...);
```

- tablo_adı: Giriş yapılacak tablo adı
- (kolon1, kolon2, kolon3, …): Giriş yapılacak kolon isimleridir. Insert işleminde otomatik artan ve NULL kabul eden kolonlar dışında kolonlara giriş yapılması zorunludur.
- VALUES (değer1, değer2, değer3, …): Kolon listesine girilen kolonlara verilecek değerlerdir. Buradaki değerler sırası ile kolon listesindeki kolonlarla eşleştirilir.

Tek Satır Veri Girişi

```
INSERT INTO Musteriler (Ad, Soyad, Telefon, Adres)
VALUES ('Sait', 'ORHAN', '1234567890', 'saitorhan.com')
```

(1 row affected) mesajı işlemin başarılı olduğunu ve bir satır

verinin girildiğini belirtir.

Varsayılan Değer Girme

Insert anında değeri hesaplanıp girilecek kolonlarda tablo tasarlanırken “Default Value or Binding” özelliğine varsayılan değeri girilir. Örneğin Müşteriler tablosunda “Kayıt Tarihi” kolonu kayıt anındaki zamanı alır. Her defasında girmek yerine kolonun varsayılan değerine “GETDATE()” fonksiyonu yazılırsa tarih değerini alıp kolon değerine girer.

```
INSERT INTO Musteriler (Ad, Soyad, Telefon, Adres,
KayitTarihi)
VALUES ('Sait', 'ORHAN', '1234567890', 'saitorhan.com',
DEFAULT)
```

Sorgunun VALUES parametrelerinden DEFAULT değeri kolonun varsayılan değeri olan GETDATE() fonksiyonunu çağırarak değerini KayitTarihi kolonuna atar.

Birden Fazla Kayıt Girmek

Insert işlemi tek satır veri girmeyi sağladığı gibi aynı sorguda birden fazla satır veri girmeyi de destekler. Girilecek değerler virgül (,) ile ayırarak girilebilir.

```
INSERT INTO Musteriler (Ad, Soyad, Telefon, Adres,
KayitTarihi)
VALUES
('Sait', 'ORHAN', '1234567890', 'saitorhan.com', DEFAULT),
('Bilal', 'ORHAN', '1234567890', 'saitorhan.com', DEFAULT),
('Said Nur', 'Yağmahan', '1234567890', 'saitorhan.com',
DEFAULT),
('Suheyb', 'Yağmahan', '1234567890', 'saitorhan.com', DEFAULT)
```

Bu sorgu sonucunda dört satırın girildiğini belirten “(4 rows affected)” mesajı gösterilecektir.

IDENTITY (Otomatik Artan Değer) Kolonuna Değer Girme

Otomatik artan olan kolona veri girildiğinde aşağıdaki gibi hata alırız.

```
Msg 544, Level 16, State 1, Line 3
Cannot insert explicit value for identity column in table
'Musteriler' when IDENTITY_INSERT is set to OFF.
```

Otomatik artan kolonuna değer girebilmek için ilgili tablo için IDENTITY_INSERT değerinin ON değerine alınması gerekmektedir. İşlem bittikten sonra değeri OFF değerine ayarlamayı unutmayınız.

```
SET IDENTITY_INSERT Musteriler ON

INSERT INTO Musteriler (Id,Ad, Soyad, Telefon, Adres,
KayitTarihi)
VALUES
(17,'Sait', 'ORHAN', '1234567890', 'saitorhan.com', DEFAULT),
(18,'Bilal', 'ORHAN', '1234567890', 'saitorhan.com', DEFAULT),
(19,'Said Nur', 'Yağmahan', '1234567890', 'saitorhan.com',
DEFAULT),
(20,'Suheyb', 'Yağmahan', '1234567890', 'saitorhan.com',
DEFAULT)

SET IDENTITY_INSERT Musteriler OFF
```

Sorgunun ilk satırında IDENTITY_INSERT değeri açılmış, son satırında da kapatılmıştır.

SELECT Sonucunun Başka Tabloya INSERT Edilmesi

Çoklu INSERT işlemi elle girilen değerler olabileceği gibi SELECT sorgusunun sonucu da olabilir. INSERT komutunda kolon listesi yazıldıktan sonra sırası eşleşecek şekilde yazılan SELECT sorgusunun sonucu INSERT komutuna iletilir.

```
INSERT INTO Musteriler2 (Ad, Soyad, Telefon, Adres,
KayitTarihi)

SELECT Ad, Soyad, Telefon, Adres, KayitTarihi
```

```
FROM Musteriler
```

Örnek sorguda , Musteriler tablosundan alınan kayıtlar Musteriler2 tablosuna INSERT edilmektedir.

INSERT İle Girilen Değerlerin Gösterilmesi

INSERT komutu ile kullanılacak OUTPUT parametresi ile girilen değerler gösterilebilir. Normal şartlarda INSERT işlemi sonucunda eklenen satır sayısı gösterilirken OUTPUT parametresi ile değerler ekrana gösterilebilir.

```
INSERT INTO Musteriler (Ad, Soyad, Telefon, Adres,
KayitTarihi)
OUTPUT inserted.Id, inserted.Ad, inserted.Soyad,
inserted.Telefon, inserted.Adres, inserted.KayitTarihi
VALUES
('Sait', 'ORHAN', '1234567890', 'saitorhan.com', DEFAULT),
('Bilal', 'ORHAN', '1234567890', 'saitorhan.com', DEFAULT),
('Said Nur', 'Yağmahan', '1234567890', 'saitorhan.com',
DEFAULT),
('Suheyb', 'Yağmahan', '1234567890', 'saitorhan.com', DEFAULT)
```

Sorgunun ikinci satırında gösterilen OUTPUT parametresi kendisine verilen kolonları yeni gelen kayıtlardan seçerek gösterir. “inserted” mevcut sorguda eklenen kayıtları tutan sanal bir

tablodur.

SQL Server Ranking Fonksiyonları

Ranking fonksiyonları sıralama işlevinde satırlara sıra numarası vermek için kullanılan fonksiyonlardır. Kullanılan fonksiyona göre bazı sıra numaraları aynı olabilir.

ROW_NUMBER

Sorgu sonucunu numaralandırır. Kullanılan yönteme göre bütün sorgu sonucunu tek grup olarak algılayıp numaralandırabileceği gibi her grubu kendi içinde tekrar birden başlayarak numaralandırabilir.

Kullanım Şekli

```
ROW_NUMBER ( )
    OVER ( [ PARTITION BY value_expression , ... [ n ] ]
order_by_clause )
```

Örnekler

Basit Örnek

Aşağıdaki kod sistem veri tabanlarını listeler. “ROW_NUMBER() OVER(ORDER BY name ASC) AS Row#” komut parçası veri tabanı isimlerini küçükten büyüğe sıralayarak numarasını verir.

```
SELECT
  ROW_NUMBER() OVER(ORDER BY name ASC) AS Row#,
  name, recovery_model_desc
FROM sys.databases
WHERE database_id < 5;
```

Row#	**name**	**recovery_model_desc**
1	master	SIMPLE
2	model	FULL
3	msdb	SIMPLE
4	tempdb	SIMPLE

“[PARTITION BY value_expression , … [n]]” kısmı kullanıldığında value-expression değerlerine göre sonucu gruplara ayırıp her grubu kendi içerisinde birden başlayarak numaralandırır. Gruplama için seçilen değer değiştiğinde sayaç tekrar bire alınır. Örneğin aşağıdaki sorguda işlemi “recovery_model_desc” kolonuna göre gruplama işlemi yapılmış.

```
SELECT
  ROW_NUMBER() OVER(PARTITION BY recovery_model_desc ORDER BY
name ASC)
    AS Row#,
  name, recovery_model_desc
FROM sys.databases WHERE database_id < 5;
```

Row#	**name**	**recovery_model_desc**
1	model	FULL
1	master	SIMPLE
2	msdb	SIMPLE
3	tempdb	SIMPLE

Sonuçta da gördüğümüz gibi recovery_model_desc kolonu “FULL” değeri için 1 verilmişken, “SIMPLE” değeri de kendi içinde numaralandırılmıştır.

Satış Temsilcilerinin Satış Miktarlarının Alınması

Başka bir örnek olarak aşağıdaki sorgu, satış temsilcilerinin yıla göre satış miktarını sorguluyor. Sorguyu yaparken de “ROW_NUMBER() OVER(ORDER BY SalesYTD DESC)” kısmı ile satış miktarlarını azalan sırada sıralayıp numaralandırmaktadır.

```
USE AdventureWorks2012;
```

```
GO
SELECT ROW_NUMBER() OVER(ORDER BY SalesYTD DESC) AS Row,
    FirstName, LastName, ROUND(SalesYTD,2,1) AS "Sales YTD"
FROM Sales.vSalesPerson
WHERE TerritoryName IS NOT NULL AND SalesYTD <> 0;
```

Row	FirstName	LastName	Sales YTD
1	Linda	Mitchell	4251368,54
2	Jae	Pak	4116871,22
3	Michael	Blythe	3763178,17
4	Jillian	Carson	3189418,36
5	Ranjit	Varkey Chudukatil	3121616,32
6	José	Saraiva	2604540,71
7	Shu	Ito	2458535,61
8	Tsvi	Reiter	2315185,61
9	Rachel	Valdez	1827066,71
10	Tete	Mensa-Annan	1576562,19
11	David	Campbell	1573012,93
12	Garrett	Vargas	1453719,46
13	Lynn	Tsoflias	1421810,92
14	Pamela	Ansman-Wolfe	1352577,13

ROW_NUMBER

ROW_NUMBER’IN PARTITION İLE KULLANILMASI

Yukarıdaki sorguyu bu sefer satış temsilcilerini bölgelerine göre gruplayıp kendi grupları içerisinde sıralamak için değiştirirsek;

```
USE AdventureWorks2012;
GO
SELECT FirstName, LastName, TerritoryName, ROUND(SalesYTD,2,1)
AS SalesYTD,
ROW_NUMBER() OVER(PARTITION BY TerritoryName ORDER BY SalesYTD
DESC)
  AS Row
```

```
 FROM Sales.vSalesPerson
 WHERE TerritoryName IS NOT NULL AND SalesYTD <> 0
 ORDER BY TerritoryName;
```

FirstName	LastName	TerritoryName	SalesYTD	Row
Lynn	Tsoflias	Australia	1421810,92	1
José	Saraiva	Canada	2604540,71	1
Garrett	Vargas	Canada	1453719,46	2
Jillian	Carson	Central	3189418,36	1
Ranjit	Varkey Chudukatil	France	3121616,32	1
Rachel	Valdez	Germany	1827066,71	1
Michael	Blythe	Northeast	3763178,17	1
Tete	Mensa-Annan	Northwest	1576562,19	1
David	Campbell	Northwest	1573012,93	2
Pamela	Ansman-Wolfe	Northwest	1352577,13	3
Tsvi	Reiter	Southeast	2315185,61	1
Linda	Mitchell	Southwest	4251368,54	1
Shu	Ito	Southwest	2458535,61	2
Jae	Pak	United Kingdom	4116871,22	1

ROW_NUMBER PARTITION

Sonuç ekranında da göründüğü gibi bölge adı (TerritoryName) değiştikçe sayaç bire eşitleniyor.

RANK

Çalışması ve kullanımı ROW_NUMBER gibidir. Farklı olarak ROW_NUMBER her bir satırı artan sırada numaralandırırken (1,2,3,4,5) RANK eşit değere sahip satırları aynı numara ile numaralandırmaktadır (1,2,2,3,4) dolayısı ile aynı değerleri dönderebilir.

Kullanım Şekli

```
RANK ( ) OVER ( [ partition_by_clause ] order_by_clause )
```

Eğer bir değer tekrar ederse sonra ki değer tekrar edenlerin sayısı toplamı kadar olacaktır. Örneğin 1, 2, 2, 2, 4, 5 vb. İki değerinden sonra 4 değerinin gelmesinin nedeni 4 değerinin gerçekten dördüncü sırada olmasıdır ve kendisinden önce bir tane birinci ve üç tana ikinci eleman bulunmaktadır. Bu özelliğinin sonucu olarak RANK fonksiyonu ardışık numaralar döndürmeyebilir.

Aşağıdaki örnek, maaşlarına göre sıralanmış ilk on çalışanı döndürmektedir. PARTITION BY deyimi belirtilmediğinden, RANK işlevi sonuç kümesindeki tüm satırlara uygulandı.

```
USE AdventureWorks2012
SELECT TOP(10) BusinessEntityID, Rate,
       RANK() OVER (ORDER BY Rate DESC) AS RankBySalary
FROM HumanResources.EmployeePayHistory AS eph1
WHERE RateChangeDate = (SELECT MAX(RateChangeDate)
                        FROM HumanResources.EmployeePayHistory
AS eph2
                        WHERE eph1.BusinessEntityID =
eph2.BusinessEntityID)
ORDER BY BusinessEntityID;
```

BusinessEntityID	Rate	RankBySalary
1	125,50	1
2	63,4615	4
3	43,2692	11
4	29,8462	28
5	32,6923	22
6	32,6923	22
7	50,4808	6
8	40,8654	14
9	40,8654	14
10	42,4808	13

RANK

Aşağıdaki örnek sorguda da ürünleri stok durumlarına ve konumlarına göre sıralar. Ürünleri LocationID değerine göre gruplayıp Quantity değerine göre azalan sırada listeler. Listenin ilk iki ürünü olan 494 ve 495 numaralı ürünlerin aynı RANK değerine sahip olduğuna dikkat edin. Sonra gelen değer tekrar eden iki 1'den dolayı üç olmuştur. Bir diğer nokta PARTITION BY ile kullanılan LocationID değeri değiştiğinde sayacın bire eşitlendiğine dikkat edin.

```
USE AdventureWorks2012;
GO
SELECT i.ProductID, p.Name, i.LocationID, i.Quantity
    ,RANK() OVER
    (PARTITION BY i.LocationID ORDER BY i.Quantity DESC) AS
Rank
FROM Production.ProductInventory AS i
INNER JOIN Production.Product AS p
    ON i.ProductID = p.ProductID
WHERE i.LocationID BETWEEN 3 AND 4
ORDER BY i.LocationID;
GO
```

ProductID	Name	LocationID	Quantity	Rank
494	Paint - Silver	3	49	1
495	Paint - Blue	3	49	1
493	Paint - Red	3	41	3
496	Paint - Yellow	3	30	4
492	Paint - Black	3	17	5
495	Paint - Blue	4	35	1
496	Paint - Yellow	4	25	2
493	Paint - Red	4	24	3
492	Paint - Black	4	14	4
494	Paint - Silver	4	12	5

RANK PARTITION

DENSE_RANK

Kullanımı RANK ile birebir aynıdır. RANK fonksiyonundan farkı, RANK fonksiyonunda tekrar eden değerden sonra kayıtların sayısı kadar olan değer geliyorken DENSE_RANK fonksiyonunda tekrar eden değerlerden sonra ardışık değer gelmektedir. Yukarıdaki sorgunun DENSE_RANK ile yazılmış hali:

```
USE AdventureWorks2012;
GO
SELECT i.ProductID, p.Name, i.LocationID, i.Quantity
    ,DENSE_RANK() OVER
    (PARTITION BY i.LocationID ORDER BY i.Quantity DESC) AS
Rank
FROM Production.ProductInventory AS i
INNER JOIN Production.Product AS p
    ON i.ProductID = p.ProductID
WHERE i.LocationID BETWEEN 3 AND 4
ORDER BY i.LocationID;
GO
```

ProductID	Name	LocationID	Quantity	Rank
494	Paint - Silver	3	49	1
495	Paint - Blue	3	49	1
493	Paint - Red	3	41	2
496	Paint - Yellow	3	30	3
492	Paint - Black	3	17	4
495	Paint - Blue	4	35	1
496	Paint - Yellow	4	25	2
493	Paint - Red	4	24	3
492	Paint - Black	4	14	4
494	Paint - Silver	4	12	5

DENSE_RANK

SQL Server Tabloların Özetlenmesi

Veri tabanı sistemlerin sağladığı özelliklerden biri de tabloların özet bilgi halinde sunulabilmesi imkanıdır. Bu özet bilgi tekniklerinden biri de kayıtların belli kriterlere göre gruplanabilmesidir. SQL Server sisteminde gruplama işlemi **GROUP BY** komutu ile yapılabilmektedir. Gruplama işlemi sonucunda her grup için bir satır veri dönecektir. SQL Server üzerinde yapılabilecek gruplama işlemi türlerini incelemeye başlayalım.

Gruplama İşlemi Yazım Şekli

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

SELECT yazılan kolonların GROUP BY kısmında geçmesi gerekmektedir. GROUP BY ile gruplanmayan ve herhangi bir gruplama fonksiyonu (SUM, AVG vb.) ile kullanılmayan kolonlar SELECT içerisinde kullanılamazlar.

GROUP BY

Aşağıdaki sql sorgusu “Satışlar” tablosunu şehir ve ilçeye göre gruplayarak satış toplamlarını vermektedir.

```
select Sehir, Ilce, sum(Tutar) [Toplam Satış]
from Satislar
group by Sehir, Ilce
```

Sehir	Ilce	Toplam Satış
Batman	Hasankeyf	8880,00
Batman	Merkez	19988,00
İstanbul	Merkez	18203,00
İstanbul	Zeytinburnu	9355,00

GROUP BY

GROUP BY ROLLUP

Gruplama işlemi için verilen kolon listesini her adımda bir azaltarak ara toplamı verir. Bu işlem için sorguyu her defasında sondan bir kolonu NULL değeri ile değiştirerek sonucu hesaplar. Örneğin GROUP BY ROLLUP(a,b,c,d) şeklinde verilen bir gruplama sorgusunu sonuçlarını aşağıdaki sonuçları verecek şekilde oluşturur.

- a, b, c, d
- a, b, c, NULL
- a, b, NULL, NULL
- a, NULL, NULL, NULL
- NULL, NULL, NULL, NULL

Örnek bir ROLLUP sorgusu

```
select Sehir, Ilce, sum(Tutar) [Toplam Satış]
from Satislar
group by rollup (Sehir, Ilce)
--(a,b,c,d) =>
--(a,b,c,d),
--(a,b,c, NULL),
--(a, b, NULL, NULL),
--(a, NULL, NULL, NULL),
--(NULL, NULL, NULL, NULL)
```

Sehir	Ilce	Toplam Satış
Batman	Hasankeyf	8880,00
Batman	Merkez	19988,00
Batman	NULL	28868,00
İstanbul	Merkez	18203,00
İstanbul	Zeytinburnu	9355,00
İstanbul	NULL	27558,00
NULL	NULL	56426,00

GROUP BY ROLLUP

GROUP BY CUBE

GROUP BY ROLLUP mantığında çalışır ancak farkı olarak mümkün olan bütün kombinasyonlar için ara toplam hesabı yapar. Örneğin GROUP BY CUBE(a, b) sorgusu için aşağıdaki kombinasyonların ara toplam hesaplamalarını yapar.

- a,b
- a, NULL
- NULL, b
- NULL, NULL

Örnek bir GROUP BY CUBE sorgusu

```
select Sehir, Ilce, sum(Tutar)  [Toplam Satış]
from Satislar
group by cube (Sehir, Ilce)
-- (a,b) =>
--(a,b),
--(NULL, b),
--(a, NULL),
```

```
--(NULL, NULL)
```

Sehir	Ilce	Toplam Satış
Batman	Hasankeyf	8880,00
NULL	Hasankeyf	8880,00
Batman	Merkez	19988,00
İstanbul	Merkez	18203,00
NULL	Merkez	38191,00
İstanbul	Zeytinburnu	9355,00
NULL	Zeytinburnu	9355,00
NULL	NULL	56426,00
Batman	NULL	28868,00
İstanbul	NULL	27558,00

GROUP BY CUBE

GROUP BY GROUPING SETS

Bazı durumlarda sorguyu birden fazla şarta göre gruplamak gerekiyor. group by GROUPING sets sorgusu parametre olarak aldığı gruplama şartlarını union all ile birleştirerek tek sonuç kümesi olarak döner.

Örneğin aşağıda yazılan sql sorgusu yukarıda açıkladığımız rollup ve cube sorgularını birleştirerek sonuç dönüyor.

```
select Sehir, Ilce, SUM(Tutar)  [Toplam Satış]
from Satislar
group by GROUPING sets(ROLLUP(Sehir, Ilce), cube(Sehir, Ilce))
--rollup ve cube işlemlerini union all ile birleştirir
```

Sehir	Ilce	Toplam Satış
Batman	Hasankeyf	8880,00
NULL	Hasankeyf	8880,00
Batman	Merkez	19988,00
İstanbul	Merkez	18203,00
NULL	Merkez	38191,00
İstanbul	Zeytinburnu	9355,00
NULL	Zeytinburnu	9355,00
NULL	NULL	56426,00
Batman	Hasankeyf	8880,00
Batman	Merkez	19988,00
Batman	NULL	28868,00
İstanbul	Merkez	18203,00
İstanbul	Zeytinburnu	9355,00
İstanbul	NULL	27558,00
NULL	NULL	56426,00
Batman	NULL	28868,00
İstanbul	NULL	27558,00

GROUP BY GROUPING SETS

GROUP BY ()

GROUP BY GROUPING SETS, parametresine () şeklinde verilen parametre (NULL, NULL, NULL …) şeklinde tablonun genel toplamını verir.

Genel Notlar

SELECT

- AVG, SUM gibi fonksiyonlar select içerisinde kullanıldığında sonuç olarak ilgili gruba ait hesaplamayı döner
- fonksiyon(DISTING kolon) şeklinde kullanılan fonksiyon sadece farklı değerleri dikkate alarak hesaplama yapar. Tekrar eden değerlerden sadece birini alır.

WHERE

WHERE ile verilen şarta uymayan kayıtlar sorguda dikkate alınmayacaktır.

HAVING

Gruplar üzerinde sorgu oluşturmak için having sorgusu kullaılır. Örneğin satış toplamı 10000’den büyük olan değerleri almak için:

```
select Sehir, Ilce, SUM(Tutar)  [Toplam Satış]
from Satislar
group by GROUPING sets(ROLLUP(Sehir, Ilce), cube(Sehir, Ilce))
having sum(Tutar) > 50000
```

NULL Değerler

SQL Server bütün NULL eşit kabul edilip tek grup altında toplanacaktır.