SQL Server 2008 ile Gelen Yeni Veri Tipleri-1
(Uzaysal Veri Tipleri)

Özet : *Bu makalemizde SQL Server 2008 'in yeni,veri tiplerinden olan Uzaysal Veri Tiplerini (GEOMETRY ve GEOGRAPHY) ve kullandığı fonksiyonları,örnek sorgularla inceleyeceğiz.*

Makale :

SQL Server 2008 ile birlikte gelen birçok yenilikten biri de,yeni veri tipleridir.SQL Server 2008'de daha önceki sürümlerde kullandığımız *int,varchar,float* gibi veri tiplerine ek olarak sınırlarımızı daha da genişleten yeni veri tipleri karşımıza çıkmaktadır.

SQL Server 2008 ile gelen;bu yeni veri tiplerini aşağıdaki gibi bir gruplamaya sokabiliriz.(Bu makalemizde ilk grup olan Uzaysal Veri Tipleri üzerinde duracağız)

- *Uzaysal Veri Tipleri:* Bu veri tipleri GEOMETRY ve GEOGRAPHY veri tiplerini içermektedir. Bu verilere de Uzaysal Veriler denmektedir.

- *Yeni Date ve Time Veri Tipleri:* Bu veri tipleri,esnek Date ve Time verilerinin tutulmasını sağlamaktadır.Örneğin;1753 yılından önceki yılların yada 3.33 milisaniye den daha kısa zaman aralıklarının,veri tabanında saklanmasına imkan sağlıyorlar.

- *Hierarchyid Veri Tipleri :* Hiyerarşik verilerin tutulduğu tipler.Örneğin; çalışan => müdür ilişkisinin tutulmasına imkan sağlayan veri tipleridir.

- *FileStream Desteği :* Büyük objelerin dosyalarda tutulması ve bu dosyaların database ile entegrasyonun sağlanması ise veri tiplerine getirilen bir başka kullanışlı yeniliktir.

Şimdi,Uzaysal Veri Tiplerine ve sağladığı imkanlara bakalım.

1)SQL Server 2008'de Uzaysal Veri Tipleri :

İstanbul'un enlem ve boylam bilgisini tutmak için (28:58E ,41:01N);veri tabanında float bir veri tanımlaması yapmış olalım.Bunun yanında enlem ve boylamla ilişkili;alansal kodları da,koordinatlarıyla bağlantılı olarak veri tabanımızda tutmak istediğimizde,gene float tiplerde veriler tanımlayarak,bir veri tabanı yapısına ulaşmış olalım.Oluşturmuş olduğumuz bu karmaşık veri tabanını başka bir uygulamanın kullanması gerektiğinde(örneğin bir web servisi),karmaşıklığından dolayı,işin içinden çıkmak oldukça zor olacaktır.Tüm bu uzaysal formatların daha da anlaşılabilir olarak tanımlanması için belirli standartların olması gerekliliği ortaya çıkmaktadır.

Bu amaçla;Open Geospatial Consortium (OGC) adlı organizasyon,açık GIS(Geographical Information System)yazılım standartlarını hazırlamaktadır.(www.opengeospatial.org adresinden ayrıntılı bilgi alınabilir)İşte SQL Server uzaysal bilgileri de ,OpenGis standartlarını kullanmaktadırlar.

SQL Serverda uzaysal veriler için 2 tane Veri Tipi mevcutdur.

a) GEOMETRY ;harita bilgilerini 2 boyutlu olarak tutmaktadır.(X ve Y düzleminde)

b) GEOGRAPHY ; verileri dünya yüzeyi ile ilişkili olarak tutmaktadır.

Şimdi bu 2 uzaysal veri formuna ayrıntılı olarak bakabiliriz.

a)GEOMETRY Tipi : GEOMETRY data tipi .NET CLR (Common Language Runtime) destekli bir SQL Server tipidir.2 boyutlu koordinat düzleminde;nokta,doğru,çember,çokgen gibi geometrik cisimlerin X ve Y eksenindeki koordinat bilgilerini tutar.

SQL Server'da;uzaysal veriler binary olarak tutulmaktadır ve text hale dönüştürülmesinde OGC tarafından tanımlanmış WKT (Well Konown Text) formatını kullanmaktadır.WKT 'lerin temsil etdiği uzaysal verilere,şunlar örnek olarak gösterilebilirler.

Geometrik Tip	WKT Gösterimi	Açıklama
Point	POINT (10 15)	Bir nokta
Multipoint	MULTIPOINT (10 10, 50 50)	İki Nokta
LineString	LINESTRING (10 10,20 20,31 35)	Üç noktası verilmiş doğru
Polygon	POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))	Beş noktadan oluşan çokgen

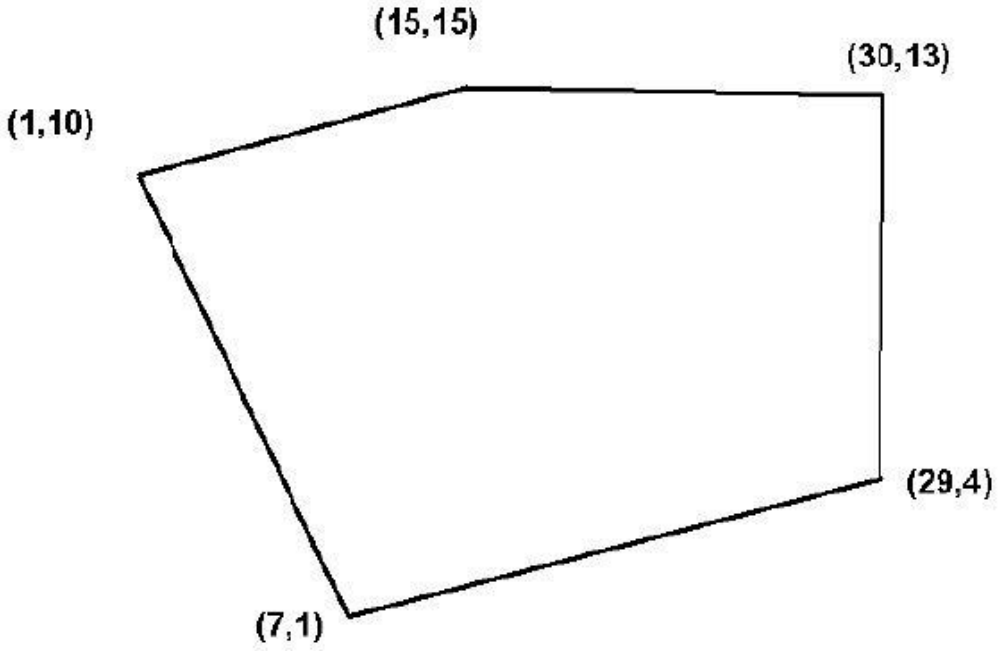
Bu yukardaki verilerin,temsili için *STAsText()* fonksiyonu kullanılmaktadır.Bu fonksiyon dışında 60'dan fazla fonksiyon GEOMETRY tipinin işlemlerinde kullanılırlar.(Aynı zamanda GEOGRAPHY tipi için de kullanılabilirler) Bu fonksiyonlardan bazıları şunlardır.

Fonksiyon	Açıklama
STAsText	Tipin değerini WKT formatında döndürür.
STGeometryType	Tipin tutduğu değerin bilinen ismini(WKT) döndürür.
STGeomFromText	WKT formatındaki değerin,geometrik değerini oluşturur.
STArea	Tipin tutduğu değerin,toplam alanı döndürür.
STSrid	Tutulan Geometrik tipin;Uzaysal Referans ID(SRID) sini döndürür.
STTouches	İki geometrik cisim komşu ise 1 değerini komşu değilse 0 değerini döndürür.
STWithin	Bir noktanın bir geometrik cisim içerisinde olması durumunda 1 değerini yoksa 0 değerini döndürür.
STDistance	Bir nokta ile bir geometrik cisim arasındaki uzaklığı döndürür.
GeomFromGML	Geography Markup Language (GML) formatında,verilen geometrik cismi döndürür.
AsGML	GML formatında,geometrik değeri döner.

Bütün geometrik tipler,bir SRID'ye sahiptir ve bu fonksiyonlar çalışırken SRID 'ye göre işlemlerini gerçekleştirir.(GEOMETRY data tipinde kullanılan bütün fonksiyonların ayrıntılı bilgisine,SQL Server Books Online'dan erişebilir.)

Buraya kadar SQL Server GEOMETRY *tipleri hakkında sizlere, temel teorik bilgileri vermeye çalıştım.Bundan sonraysa SQL Server'da bu tipleri ve fonksiyonları örnek sorgularla inceleyeceğiz.*

Öncelikle hayali kordinatlara sahip çokgensel bir alanımızın olduğunu varsayalım.



Alan Kodları İçin Hayali Alanımız

İlk olarak Geometrik Verilerimizin tutulacağı AlanKodlari adlı tablomuzu oluşturuyoruz.

```
CREATE TABLE AlanKodlari
(ID INT PRIMARY KEY,
AKodununGeometrisi GEOMETRY NOT NULL,
AKodununTexti AS AKodununGeometrisi.STAsText()
)
```

Buradaki AkodununGeometrisi kolonu ,Alan Kodlarımızı tutacak.

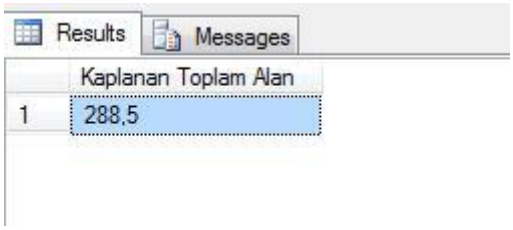
Şimdi yukardaki şekildeki noktalarımızı,KodunDatasi adlı değişkene giriyoruz ve ardından da AlanKodlari tablosunun,1 numaralı ID li kayıdının noktasal değerlerini veriyoruz.

```
DECLARE @KodunDatasi GEOMETRY
SET @KodunDatasi = geometry::STGeomFromText('POLYGON((1 10,
15 15, 30 13, 29 4, 7 1, 1 10))', 0)
INSERT INTO AlanKodlari
VALUES ('1',GEOMETRY::STGeomFromText('POLYGON((1 10,
15 15, 30 13, 29 4, 7 1, 1 10))', 0))
```

Buradaki `STGeomFromText` fonksiyonu WKT alır ve SQL Server'ın storage engine'ninde binary olarak saklar.Bu örnek için çokgenin taslağı bir Alan Kodu olarak saklanmaktadır.

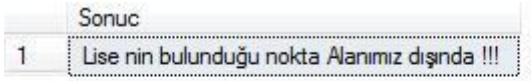
Şimdi de vermiş olduğumuz noktasal kordinatların oluşturacağı çokgensel şeklin toplam kapladığı alanı `STArea()` fonksiyonu ile hesaplayalım.

```
DECLARE @KodunDatasi GEOMETRY;
SET @KodunDatasi=(SELECT AKodununGeometrisi from AlanKodlari where
ID=1)
SELECT @KodunDatasi.STArea() as 'Kaplanan Toplam Alan'
```



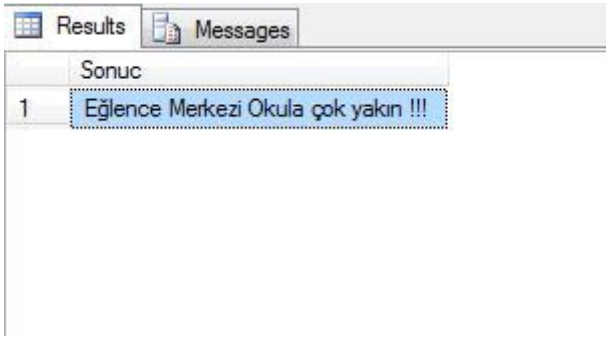
Örnek olarak bu alan içinde,hayali bir noktaya bir Lise kurulacak olsun.Bu;lisenin bulunduğu noktanın;alanımız içinde olup olmadığını `STWithin()` fonksiyonunu kullanarak bulabiliriz.

```
DECLARE @Lise GEOMETRY
SET @Lise=GEOMETRY::STGeomFromText('POINT(23 32)',0);
IF (@Lise.STWithin(@KodunDatasi)=1)
SELECT 'Lise nin bulunduğu nokta Alanımız içerisinde' as 'Sonuc'
ELSE
SELECT 'Lise nin bulunduğu nokta Alanımız dışında !!!' as 'Sonuc'
```



Şimdi de sorgumuzu biraz daha geliştirelim;bir eğlence merkezinin açılma şartının değerlendirilmesi için,Liseye olan uzaklığını `STDistince()` fonksiyonunu kullanarak bulduktan sonra,değerlendirmenin yapılmasını sağlayalım.(1 KM'den küçükse kurulamasın kriteri olsun mesela)

```
DECLARE @EglenceMerkezi GEOMETRY
DECLARE @Lise GEOMETRY
SET @Lise=GEOMETRY::STGeomFromText('POINT(23 32)',0);
SET @EglenceMerkezi=GEOMETRY::STGeomFromText('POINT(12 8)',0);
IF (@EglenceMerkezi.STDistance(@Lise)<1.0)
SELECT 'Eğlence Merkezi Okula çok yakın !!!' as 'Sonuc'
ELSE
SELECT 'Eğlence Merkezi Okula yakın değil' as 'Sonuc'
```



Şimdi de `STTouches()` fonksiyonunu kullanarak;doğru şeklinde olan ve kordinatları verilen bir nehirin,alanımız ile sınır olup olmadığını bulalım.

```
DECLARE @Nehir GEOMETRY
SET @Nehir=GEOMETRY::STGeomFromText('LINESTRING(.5 12, 1 10, 1 8, .5 3, 0
0)',0)
SELECT @Nehir.STTouches(@KodunDatasi)
```

Çıkan sonucun 1 olması,bize nehirin alanımızla sınır olduğunu gösteriyor.

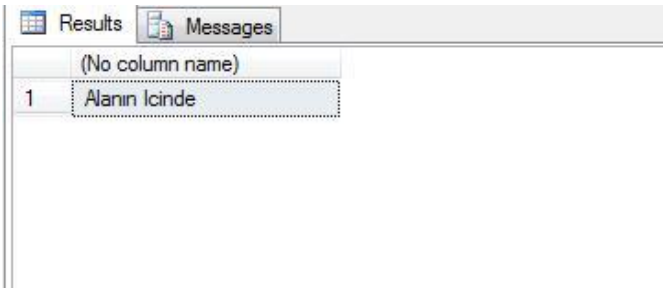
SQL Server 2008'in fonksiyonlarının,iletişimde bulunduğu bir başka data grubu ise XML datalardır.XML datalar için Geometrik veriler,Geography Markup Language (GML) aracılığıyla tanımlanabilir.

GML'in SQL Server 2008 de çalışma şeklini incelemek amacıyla 2 tane girdi alan(XML ve integer),`AlanKodununIcindeMi` adlı bir store procedure yazalım.XML girdimizi bir noktayı GML formatında tanımlamak için oluştururken,integer girdimizi Alan Kodlarını karşılaştırmak için oluşturuyoruz.

```
CREATE PROCEDURE AlanKodununIcindeMi(@KodGML as xml,@AlanKodu as int)
AS
BEGIN
DECLARE @KodDatasi GEOMETRY;
SET @KodDatasi=(SELECT AKodununGeometrisi from AlanKodlari where
ID=@AlanKodu)
DECLARE @NoktaGML GEOMETRY;
SET @NoktaGML=GEOMETRY::GeomFromGml(@KodGML,0)
IF (@NoktaGML.STWithin(@KodDatasi)=1)
SELECT 'Alanın Icinde'
ELSE
SELECT 'Alanın Dışında'
      END
```

Daha sonra da yazmış olduğumuz bu Store Procedurümüzü test etmek için (10 10) koordinatlarında bir nokta oluşturalım ve bu noktanın alanın içinde olup olmadığını, 'AlanKodununIcindeMi' adlı store procedurümüzü kullanarak bulmuş olalım.

```
DECLARE @OrnekNokta XML
SET @OrnekNokta='<Point xmlns="http://www.opengis.net/gml"><pos>10
10</pos></Point>'
EXEC AlanKodununIcindeMi @OrnekNokta,1
```



Burada anlatdığım veri tipleri üzerinde işlem yapan fonksiyonların çağrılması için .NET ortamında *Microsoft.SqlServer.Server.Types* isim alanı altında ki *SqlGeometry* tipi kullanılmalıdır.

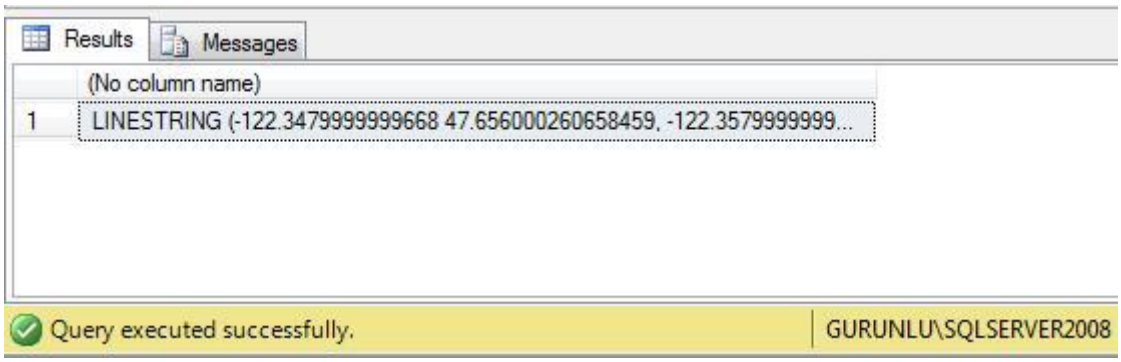
b) GEOGRAPHY Tipi :GEOGRAPHY veri tipi, GEOMETRY veri tipiyle aynı fonksiyonları kullanılır,aynı şekilde çalışır.Dünya'nın şeklinden dolayı,bu tipe geoid veri tipi denmektedir.Bundan dolayı da objeleri geodetik düzlemde ifade eder.Bu veri tipi noktaların enlemsel ve boylamsal bilgilerini GPS(Global Positioning System) kordinatlarına göre temsil eder.Gene bu tipte de her obje ,bir SRID'ye sahiptir.

Şimdi de STGeomFromText;fonksiyonunu kullanarak 1 numaralı ID'ye sahip kayıda bir Linestring'in ,2 numaralı ID'ye sahip kayıda da bir çokgenin kordinat bilgilerini girelim.

```
CREATE TABLE UzaysalVeriler
(ID int IDENTITY (1,1),
Kolon1 geography,
Kolon2 AS Kolon1.STAsText() );
GO

INSERT INTO UzaysalVeriler (Kolon1)
VALUES (geography::STGeomFromText('LINESTRING(-122.360 47.656, -122.343
47.656)', 4326));

INSERT INTO UzaysalVeriler (Kolon1)
VALUES (geography::STGeomFromText('POLYGON((-122.358 47.653, -122.348
47.649, -122.348 47.658, -122.358 47.658, -122.358 47.653))', 4326));
GO
```

STIntersects fonksiyonunu kullanarak Linestring ile Çokgenin; Linestring üzerinde ki kesiştiği noktanın kordinatlarını bulalım.

```
DECLARE @geografik1 geography;
DECLARE @geografik2 geography;
DECLARE @sonuc geography;

SELECT @geografik1 = Kolon1 FROM UzaysalVeriler WHERE ID = 1;
SELECT @geografik2 = Kolon1 FROM UzaysalVeriler WHERE ID = 2;
SELECT @sonuc = @geografik1.STIntersection(@geografik2);
SELECT @sonuc.STAsText();
```



Bu makalemizde SQL Server 2008 'in yeni tiplerinden olan Uzaysal Veri Tiplerini (GEOMETRY ve GEOGRAPHY) ve kullandığı fonksiyonlarını,örnek sorgularla inceledik.Bu makaleye bağlı makalelerde SQL Server 2008 ile gelen yeni veri tiplerini incelemeye devam edeceğiz.

Başka bir Microsoft SQL Server 2008 makalesinde görüşmek dileğiyle...

Bilgehan GÜRÜNLÜ

www.gurunlu.com
bilgehan@gurunlu.com

Kaynaklar :

-http://msdn.microsoft.com/en-us/library/
-Accelerated SQL Server 2008
-Programming Microsoft® SQL Server 2008 (Ms-Press)

SQL Server 2008 ile Gelen Yeni Veri Tipleri-2
(Date ve Time Veri Tipleri)

Özet : *Bu makalemizde SQL Server 2008 'in yeni veri tiplerinden olan Date-Time Veri Tiplerini ve yeni sistem fonksiyonlarının kullanımlarını,örnek sorgularla inceleyeceğiz.*

Makale :

SQL Server 2008 ile birlikte gelen birçok yenilikten biri de,yeni veri tipleridir.SQL Server 2008'de daha önceki sürümlerde kullandığımız *int,varchar,float.datetime* gibi veri tiplerine ek olarak sınırlarımızı daha da genişleten yeni veri tipleri karşımıza çıkmaktadır.

SQL Server 2008 ile gelen;bu yeni veri tiplerini aşağıdaki gibi bir gruplamaya sokabiliriz.(Bu makalemizde ikinci grup olan Date ve Time Veri Tipleri üzerinde duracağız)

- *Uzaysal Veri Tipleri:* Bu veri tipleri GEOMETRY ve GEOGRAPHY veri tiplerini içermektedir. Bu verilere de Uzaysal Veriler denmektedir.

- *Yeni Date ve Time Veri Tipleri:* Bu veri tipleri,esnek Date ve Time verilerinin tutulmasını sağlamaktadır.Örneğin;1753 yılından önceki yılların yada 3.33 milisaniye den daha kısa zaman aralıklarının,veri tabanında saklanmasına imkan sağlıyorlar.

- *Hierarchyid Veri Tipleri :* Hiyerarşik verilerin tutulduğu tipler.Örneğin; çalışan => müdür ilişkisinin tutulmasına imkan sağlayan veri tipleridir.

- *FileStream Desteği :* Büyük objelerin dosyalarda tutulması ve bu dosyaların database ile entegrasyonun sağlanması ise veri tiplerine getirilen bir başka kullanışlı yeniliktir.

Şimdi;yeni Date ve Time Veri Tiplerine ve sağladığı imkanlara bakalım.(Bir önceki bağlı makalemizde 1.grup olan Uzaysal Veri Tiplerini incelemiştik)

2)SQL Server 2008'de Yeni Date ve Time Veri Tipleri : SQL Server 2008 öncesi;SQL Server sürümlerinde zamansal datalarımızı datetime veya smalldatetime veri tipleri üzerinde tutuyorduk.Bu zamansal veri tiplerini kullanırken de bazı kısıtlamalar ile karşı karşıyaydık.Örneğin datetime veri tipimizin kapsadığı tarih aralığı;1 Ocak 1753 'den başlayıp 31 Aralık 9999 'a kadar gidebiliyordu. 1753 yılından önceki tarihlerin tutulmasını gerektiren bir uygulama geliştireceğimiz zamanda bu veri tiplerini daha farklı veri tipleri üzerinde tutmamız gerekmekteydi.(Örneğin bir müzede ki tarihi eserlerin verilerinin tutulduğu bir veri tabanına 1753 öncesi tarihleri nasıl girebilirdiniz ?) Datetime tipiyle ilgili bir başka kıstlamamız ise 3.33 milisaniyeden daha hassas zaman dilimlerindeki verileri,datetime tipiyle tutamayacak olmamızdı,bu durum özellikle finansal yada bilimsel uygulamaların veri tabanlarını tasarlayanlar için zorluklardan biriydi.

Zamansal veriler üzerinde ki karşılaşılan bu zorluklara ,SQL Server 2008 ile gelen,yeni Date ve Time tipleriyle çözüm bulabiliyoruz artık.

Yeni Date ve Time Veri Tiplerimiz: SQL Server 2008; DATE, TIME, DATETIMEOFFSET,

ve DATETIME2 olmak üzere 4 yeni tarihsel veri tipiyle karşımıza çıkmaktadır.Bu yeni tarihsel tiplerimiz sayesinde;1753 yılı gibi sınırlamaların ortadan kalkmasının yanında,nanosaniyeler ölçüsünde zamansal kayıtları tutabilme,Bölgesel Zaman Ayarlarına bağlı değişiklikleri kullanabilme imkanlarına sahip oluyoruz.

SQL Server 2008'e getirilmiş olan;bu yeni zamansal veri tipleri,diğer SQL Server veri tipleriyle ortak sorgularda kullanılabilmekte;DML (Data Manipulation Language) ve DDL(Data Definition Language),Store Procedurelerde yani SQL Server deyince aklınıza gelen bütün yapılar ile beraber sorunsuz bir şekilde çalışmaktadır.

SQL Server 2000 yada SQL Server 2005 kullanıcıları; Server Management Objets(SMO) aracılığıyla,yeni date ve time veri tiplerinin değerlerine ulaşmak istediğinde,text-string olarak değerlere ulaşabiliyorlar.(Bu da hatanın oluşmasının önüne geçiyor.)Aynı zamanda,bu yeni veri tiplerimiz OLE DB ve ODBC'ye de SQL Native Client aracılığıyla tam destek sunmaktadırlar.Visual Studio 2008'de geliştirdiğimiz ADO.Net projelerinde de bu veri tiplerine tam destek mevcutdur.

Şimdi bu 4 yeni veri tipimize ayrıntılı olarak bakmanın zamanı geldi.

a)DATE Tipi : Bu yeni zamansal veri tipimiz;Gregorian takvimini esas alarak 1 ile 9999 arası yıl değerlerini tutabilir.Diyelim ki çalışanların doğum tarihlerini yada işe başlama tarihlerini tutacak bir şema tasarlamak istiyorsunuz;eğer burada ki değerleri datetime tipinden oluşturursanız 6 byte'lık bir alan kaplayacaklardır.Fakat bu tarihsel alanınızı; yeni DATE tipinden tanımlarsanız sadece 3 byte'lık bir alan kapladığını göreceksiniz.Şuan için aradaki bu fark çok dikkat çekici olmayabilir ama tüm veri tabanınızı düşündüğünüz zaman bu yeni tipin faydasını daha açık olarak görebilirsiniz.

Verilerin kapladığı alanlarla ilişkili kaygılarınızdan dolayı; ay ve gün bilgisi için tinyints (1 byte) ,yıl için de smallint (2 byte) bir veri türü tanımlaması yapsanız bile gene toplamda 4 byte yapar ki,DATE türüne göre 1 byte fazla bir alan kaplanır.Diğer taraftan integer'ları kullanarak yaptığınız tanımlamalarda,iç fonksiyonların(DATEPART, DATENAME, DATEDIFF,DATEADD) kullanmak için ekstra çaba harcamanız gerekmesine rağmen DATE tiplerinde böyle bir ekstra çabaya gerek bile yoktur.

Buraya kadar anlatılanları küçük bir sorgu aracılığıyla görürsek daha iyi anlaşılacaktır.(Sorgumuzu yazmadan önce DATE tiplerinde tarih sırasının YYYY-MM-DD olarak gösterildiğini belirteyim.)

```
DECLARE @bugun DATE
DECLARE @yeniyil DATE
SET @bugun = SYSDATETIME()
SET @yeniyil = '2010-12-31'
SELECT @bugun as 'Bugunun Tarihi',
DATEADD(week,1,@bugun) as
'Bir Hafta Sonra Bugun',
DATEDIFF(day,@bugun ,@yeniyil) as
'Yeni Yıla Kalan Gun ',
DATALENGTH(@bugun) as 'Veri Kaç Byte ? '
```

Results | Messages

	Bugunun Tarihi	Bir Hafta Sonra Bugun	Yeni Yila Kalan Gun	Veri Kaç Byte ?
1	2010-01-01	2010-01-08	364	3

Bu yazmış olduğum sorguya dikkatlice bakanların gözüne;SYSDATETIME() fonksiyonu ,takılmıştır herhalde.Buradaki SYSDATETIME() fonksiyonu,SQL Server 2008 ile gelen yeni 5 fonksiyondan biridir.Yapmış olduğu işe gelince;zamanı DATETIME2(7) tipinde döndürmektir.(Tüm bu yeni fonksiyon ve tipler bu makalenin ilerleyen satırlarında anlatılacaktır.)

b)TIME Tipi : Yeni TIME(n) tipimiz;00:00:00.0000000 ile 23:59:59.9999999 aralığındaki zamanı saklamaktadır.Bu zamansal tipimiz 100 nanosaniye'den daha küçük zaman aralıklarını bile destekler.Buradaki "n" ifadesiyse 0'dan 7'ye kadar sayılar alarak zaman bölümlemesinde ki hassasiyeti belirlememizi sağlar.

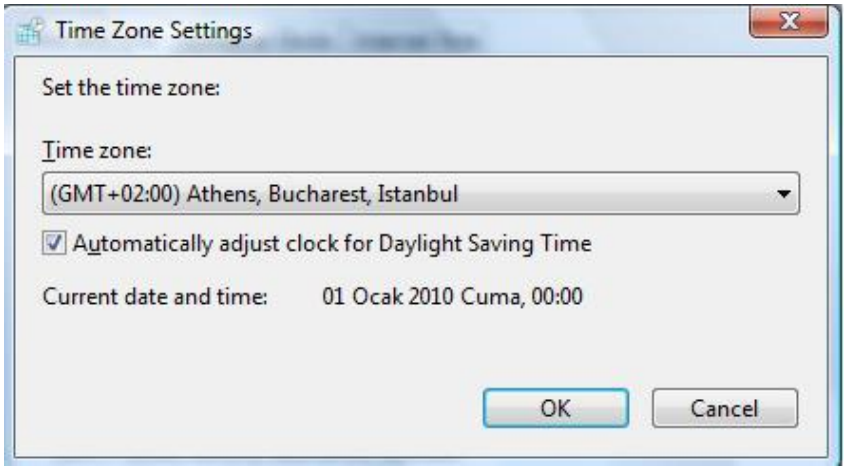
```
DECLARE @simdiki_zaman_0 time(0)
DECLARE @simdiki_zaman_1 time(1)
DECLARE @simdiki_zaman_2 time(2)
DECLARE @simdiki_zaman_3 time(3)
DECLARE @simdiki_zaman_4 time(4)
DECLARE @simdiki_zaman_5 time(5)
DECLARE @simdiki_zaman_6 time(6)
DECLARE @simdiki_zaman_7 time(7)
SET @simdiki_zaman_0=SYSDATETIME()
SET @simdiki_zaman_1=SYSDATETIME()
SET @simdiki_zaman_2=SYSDATETIME()
SET @simdiki_zaman_3=SYSDATETIME()
SET @simdiki_zaman_4=SYSDATETIME()
SET @simdiki_zaman_5=SYSDATETIME()
SET @simdiki_zaman_6=SYSDATETIME()
SET @simdiki_zaman_7=SYSDATETIME()

SELECT @simdiki_zaman_0 as 'Suan ki Zamanımız',
datalength(@simdiki_zaman_0) as 'Kaç Byte ?'
UNION ALL
SELECT @simdiki_zaman_1,datalength(@simdiki_zaman_1)
UNION ALL
SELECT @simdiki_zaman_2,datalength(@simdiki_zaman_2)
UNION ALL
SELECT @simdiki_zaman_3,datalength(@simdiki_zaman_3)
UNION ALL
SELECT @simdiki_zaman_4,datalength(@simdiki_zaman_4)
UNION ALL
SELECT @simdiki_zaman_5,datalength(@simdiki_zaman_5)
UNION ALL
SELECT @simdiki_zaman_6,datalength(@simdiki_zaman_6)
UNION ALL
SELECT @simdiki_zaman_7,datalength(@simdiki_zaman_7)
```

Results | Messages

	Suan ki Zamanımız	Kaç Byte ?
1	02:13:10.0000000	3
2	02:13:10.0000000	3
3	02:13:09.9700000	3
4	02:13:09.9710000	4
5	02:13:09.9710000	4
6	02:13:09.9710000	5
7	02:13:09.9710000	5
8	02:13:09.9710000	5

Burada da görüldüğü gibi;istenilen hassasiyete bağlı olarak verinin kapladığı alan da 3 ile 5 byte arasında değişmektedir.Ayrıca DATE tipinde olduğu gibi TIME tipi de DATEDIFF ve DATEADD gibi fonksiyonları destekler.

c)DATETIMEOFFSET Tipi : DATETIMEOFFSET(n) veri tipi;DATETIME veri tipinin bölgesel saat farklılıklarının dikkate alındığı zamansal veri tipidir.Açılımı da "date + time + time-zone offset " olarak düşünülebilir.Bulunan noktanın Coordinated Universal Time (UTC) 'a göre alacağı saat farklılıklarını esas almaktadır.(UTC ile GMT aynı şeydir.) Örneğin İstanbul için bu farklılık +2 olarak ifade edilir.



SQL Server 2008 ile bu veri tipinin kullanıma SYSDATETIMEOFFSET() adında bir sistem fonksiyonu eklenmiştir.Bu fonksiyon ile tarih ve saati Bölgesel saat farklılıklarıyla beraber gösterebilme imkanı mevcutdur.

```
SELECT SYSDATETIMEOFFSET() AS 'Şuan ki Tarih,Saat ve Time Zone Değeri'
GO
```

Results | Messages

	Şuan ki Tarih,Saat ve Time Zone Değeri
1	2010-01-01 00:00:05.9020000 +02:00

Buradaki saat farklılığı bilgileri İşletim Sistemin saatinden alınmaktadır.(Mevsimlere göre enerji tasarrufu için ,saat farklılıkları da işletim sistemine bağlı olarak değişecektir.)Burada da

DATETIMEOFFSET(n) deki hassasiyet derecesi “n” e göre tipin kaplayacağı alan,byte olarak değişiklik gösterir.

Son olarak da kullanılabilecek,bölgesel farklılık değerlerinin -14 ile +14 arasında bir değer alacağını söylemekte fayda var.

d)DATETIME2 Tipi : DATETIME2(n) veri tipimiz önceki ;SQL Server sürümlerinde ki datetime tipinin geliştirilmiş şeklidir.Aslında bu veri tipi ANSI SQL standartlarına göre TIMESTAMP olarak çağrılması gerekirken,zaten bir timestamp tipimiz SQL Serverda olduğundan dolayı geliştirici takım tarafından DATETIME2(n) olarak isimlendirilmiştir.

Bu yeni veri tipimizin,aralığı 1 Ocak 0001 yılından başlayıp,31 Aralık 9999 tarihine kadardır.Bu veri tipimiz DATETIMEOFFSET veri tipinden farklı olarak,bölgesel saat farklılıklarını barındırmaz.

DATETIME2(n) veri tipi,gene hassasiyet derecesine göre değişmek kaydıyla 6 ile 8 byte arasında bir alan kaplar.Bu veri tipimiz daha önce kullanılan datetime tipinin SQL Server 2008 deki karşılığı olarak değerlendirilip,yeni tiplere geçişte kullanılabilir.

DATETIME2(n) veri tipi sadece ANSI SQL Standartlarına göre değil, NET Datetime tipiyle de sorunsuz bir şekilde çalışır.

Buraya kadar SQL Server 2008 ile gelen 4 yeni tarihsel veri tipimizi gördük,bu makalemizin bundan sonra ki kısmında yeni gelen zamansal fonksiyonlarımızı inceleyeceğiz.

Yeni Date ve Time Sistem Fonksiyonlarımız

SQL Server 2008’de yeni eklenen date ve time veri tiplerinin;yanı sıra 5 tane de bu tiplerle ilişkili sistem fonksiyonu eklenmiştir.Bunlardan;SYSDATETIME,SYSUTCDATETIME ve SYSDATETIMEOFFSET,fonksiyonu timestamp tipinden sonuçlar döndürürken, TODATETIMEOFFSET ve SWITCHOFFSET fonksiyonları ise adlarından da anlaşılacağı gibi bölgesel zaman dilimlerine göre date ve time değerlerimizi döndürecektir.

Şimdi bu fonksiyonlara ve kullanımlarına sırasıyla bakalım.

a) SYSDATETIME Fonksiyonu :SYSDATETIME fonksiyonu; DATETIME2(7) formatında timestamp değerini bölgesel saat farklılıklarını göstermeden döndürür.

```
SELECT SYSDATETIME() as 'Şuan ki Zaman',
DATALENGTH(SYSDATETIME()) as 'Kapladıgi Alan '
```

	Şuan ki Zaman	Kapladıgi Alan
1	2010-01-01 00:01:19.6450000	8

b)SYSUTCDATETIME Fonksiyonu: SYSUTCDATETIME fonksiyonu, SYSDATETIME fonksiyonuna benzer şekilde çalışır sistem timestamp’ini UTC formatında gösterir.

```
SELECT SYSUTCDATETIME() as 'Şuan ki Zaman',
DATALENGTH(SYSUTCDATETIME()) as 'Kapladıgi Alan '
```

	Şuan ki Zaman	Kapladıgi Alan
1	2009-12-31 22:03:38.6700000	8

c)SYSDATETIMEOFFSET Fonksiyonu:Önceki 2 fonksiyona benzer şekilde sistem timestamp değerini döndürmeye ek olarak,bölgesel saat farklılık değerini de verir.(Gene DATETIME2(7) formatında)

```
SELECT SYSDATETIMEOFFSET() as 'Şuan ki Zaman',
DATALENGTH(SYSDATETIMEOFFSET()) as 'Kapladıgi Alan '
```

	Şuan ki Zaman	Kapladıgi Alan
1	2010-01-01 00:04:09.8440000 +02:00	10

d)TODATETIMEOFFSET Fonksiyonu: Bu fonksiyon yerel saat ve zaman değerlerini,date-time offset UTC değerine dönüştürerek gösterir.

```
DECLARE @OffsetYok DATETIME2
DECLARE @OffsetVar DATETIMEOFFSET
SET @OffsetYok=sysdatetime()
SET @OffsetVar=TODATETIMEOFFSET(@OffsetYok,'+02:00')
SELECT @OffsetYok AS 'Offset Olmadan',
@OffsetVar AS 'Offset Eklenirse'
```

	Offset Olmadan	Offset Eklenirse
1	2010-01-01 00:00:06.5170000	2010-01-01 00:00:06.5170000 +02:00

e) SWITCHOFFSET Fonksiyonu : SWITCHOFFSET fonksiyonu;verilen başka bölgeye ait bir zaman diliminin,bulunan zaman dilimindeki karşılığını elde edebilme imkanı sağlamaktadır.Örneğin 2 Ocak 2010 tarihinde Londra saatiyle 01:30'da başlayacak bir webinerin,İstanbul saatiyle kaçta başlayacağını bulmak istediğimizi varsayalım.

```
DECLARE @Webinerin_Zamani DATETIMEOFFSET
SET @Webinerin_Zamani ='2010-01-02 01:30 +00:00'
SELECT @Webinerin_Zamani
AS 'Webinerin Londrada Başlayacağı Zaman',
SWITCHOFFSET(@Webinerin_Zamani,'+02:00')
AS 'Webinerin Istanbulda Başlayacağı Zaman'
```

	Webinerin Londrada Başlayacağı Zaman	Webinerin Istanbulda Başlayacağı Zaman
1	2010-01-02 01:30:00.0000000 +00:00	2010-01-02 03:30:00.0000000 +02:00

Görüldüğü gibi Webinerimiz Türkiye saatine göre 03:30'de başlayacaktır.

Bu makalemizde SQL Server 2008 'in yeni veri tiplerinden olan Date-Time Veri Tiplerini ve kullandığı fonksiyonlarını,örnek sorgularla inceledik.Bu makaleye bağlı makalelerde SQL Server 2008 ile gelen yeni veri tiplerini incelemeye devam edeceğiz.

Başka bir Microsoft SQL Server 2008 makalesinde görüşmek dileğiyle...

Bilgehan GÜRÜNLÜ

www.gurunlu.com
bilgehan@gurunlu.com

Kaynaklar :

-http://msdn.microsoft.com/en-us/library/
-Accelerated SQL Server 2008
-Programming Microsoft® SQL Server 2008 (Ms-Press)

SQL Server 2008 ile Gelen Yeni Veri Tipleri-3
(Hiyerarşik Veri Tipleri)

Özet : *Bu makalemizde SQL Server 2008 'in yeni veri tiplerinden olan* Hierarchyid *Veri Tiplerini ve kullandığı fonksiyonlarını,örnek sorgularla inceleyeceğiz*

Makale :

SQL Server 2008 ile birlikte gelen birçok yenilikten biri de,yeni veri tipleridir.SQL Server 2008'de daha önceki sürümlerde kullandığımız *int,varchar,float,datetime* gibi veri tiplerine ek olarak sınırlarımızı daha da genişleten yeni veri tipleri karşımıza çıkmaktadır.

SQL Server 2008 ile gelen;bu yeni veri tiplerini aşağıdaki gibi bir gruplamaya sokabiliriz.(Bu makalemizde üçüncü grup olan Hiyerarşik Veri Tipleri üzerinde duracağız)

- *Uzaysal Veri Tipleri:* Bu veri tipleri GEOMETRY ve GEOGRAPHY veri tiplerini içermektedir. Bu verilere de Uzaysal Veriler denmektedir.

- *Yeni Date ve Time Veri Tipleri:* Bu veri tipleri,esnek Date ve Time verilerinin tutulmasını sağlamaktadır.Örneğin;1753 yılından önceki yılların yada 3.33 milisaniye den daha kısa zaman aralıklarının,veri tabanında saklanmasına imkan sağlıyorlar.

- *Hierarchyid Veri Tipleri :* Hiyerarşik verilerin tutulduğu tipler.Örneğin; çalışan => müdür ilişkisinin tutulmasına imkan sağlayan veri tipleridir.

- *FileStream Desteği :* Büyük objelerin dosyalarda tutulması ve bu dosyaların database ile entegrasyonun sağlanması ise veri tiplerine getirilen bir başka kullanışlı yeniliktir.

Şimdi;yeni Hiyerarşik Veri Tiplerine ve sağladığı imkanlara bakalım.(Önceki bağlı makalelerimizde Uzaysal Veri Tiplerini ve Date-Time Veri Tiplerini incelemiştik)

3)SQL Server 2008'de Yeni Hiyerarşik Veri Tipleri :

Hiyerarşik verilere örnek olarak;organizasyon ve yönetim tabloları,ürün katalogları,dizin yapılarının tutulma şekilleri gösterilebilir.Hiyerarşik veriler birbiriyle ilişki halinde olan üyelerden oluşan bir yapıdır.Basitçe ifade etmek gerekirse parent-child ilişkisi bir hiyerarşik yapıdır.(üst-ast ilişkisi olarak düşünülebilirler.)Parent denilen kök üyenin;türeyen kolları (child),olabileceği gibi olmaya da bilir,aynı durum child için de geçerlidir.

SQL Serverda Hiyerarşik veriler 3 farklı yolla ele alınabilir.

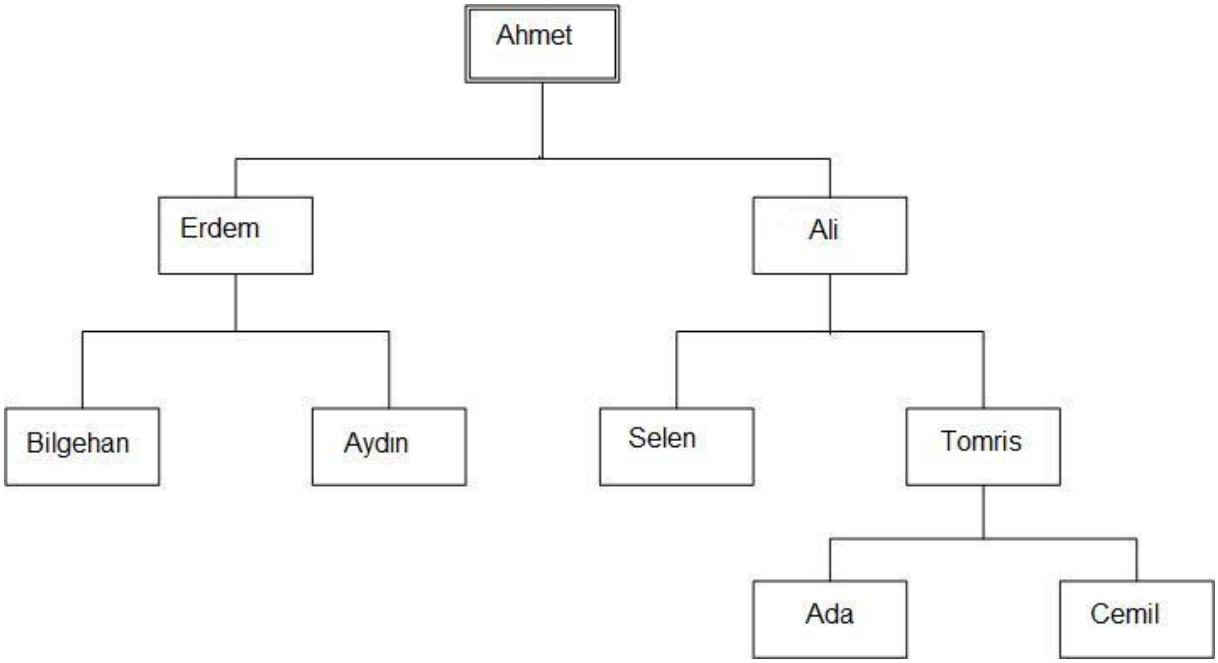
- XML yapıları kullanarak.Bu durum bütün uygulama ve kullanıcıların verilerinin XML ile ilişkilendirilmesini gerektirir.Fakat heterojen yapılar için XML veriler bazen aşırı zorlayıcı olabilirler.
- Verileri ilişkisel tablolarda tutarak ve sonra bunları SELF JOIN 'lerle birleştirerek.Bu da tabii ki karmaşık T-SQL sorguları gerektirmektedir.Ayrıca yönetim ve sürdürülebilme gibi konularda zorluklar ortaya çıkabilmektedir.
- SQL Server 2008 ile gelen yeni hierarchyid veri tipini kullanmak.

Hierarchyid veri tipi ; hiyerarşik verilerimizi tanımlayabilmenin yanında,ilişkili veriler arasında yeni gelen fonksiyonlar aracılığıyla işlem yapabilme imkanını da tanımaktadır.

Hierarchyid veri tipi;DATE ve TIME veri tipleri gibi yerel bir tip değildir,sistem tanımlı UDT(User Definied Type) bir tiptir.Bu yeni tipimiz; Microsoft.SqlServer.Types.dll altında yer almaktadır.Uygulamalarda Microsoft.SqlServer.Types aduzayı aracılıgıyla kullanabilir.

Teknik olarak Hieararchyid veri tipleri CLR UDT'dir bu yüzden SQL Serverda CLR'yi aktif yapmanız gerektiğini düşünebilirsiniz ama Hieararchyid tipler sistem tanımlı tip olduğundan dolayı CLR aktif yapmanız gerekmemektedir.

Şimdi Hierarchyid veri tipimizi daha iyi anlayabilmek için hayali bir şirketin,organizasyon şemasını oluşturalım.



Şekildeki gibi bir organizasyon şemasına sahip şirketimizin;yönetim yapısını SQL Serverda ,yeni hierarchyid tipini kullanarak saklamak isteyelim.Önce *Sirket* isimli veri tabanımızı oluşturuyoruz,daha sonra *Calisanlar* isimli tablomuzu oluşturuyoruz.

Bundan sonrasında yazacağım sorgularla,şirketimizin hiyerarşik yapısını adım adım oluşturacağım.Yukarda ki organizasyon şeması şirket yapısını daha iyi anlamanız içindir.

```
CREATE TABLE Calisanlar
(Dugum hierarchyid PRIMARY KEY CLUSTERED,
Basamak AS dugum.GetLevel() PERSISTED,
Calisan_ID INT UNIQUE,
Calisan_Isim VARCHAR(30) NOT NULL)
```

Burada ast-üst ilişkisinin SQL Server'a aktarılmasını sağlayan fonksiyonumuz GetLevel() dir.Bu fonksiyonumuz herbir basamakta ki,düğümün değerini döndürür.Örneğin kök(root) düğümün,Ahmet,Basamak değeri 0 dır.Erdem ve Ali'nin;basamak değeri 1 dir ve aşağılara indikçe basamak değerinin artmasından dolayı da en son basamaktaki,Ada ve Cemil'in,basamak değerleri 3 olarak ifade edilecektir.

Tabloyu oluşturduktan sonra kök de(root) beraberinde oluşturulmuş olur.

```
INSERT INTO Calisanlar VALUES (hierarchyid::GetRoot(),5000,'Ahmet')
```

GetRoot() fonksiyonumuzsa,root'un bulunduğu düğümdeki değeri döndürecek bir başka fonksiyonumuzdur.Burada Calisan_ID sinin değeri 5000 'den başlarken, basamaklar 0'dan başlayarak artacaktır.Bu iki ifadeyi birbirine karıştırmamamız gerekir.

Elinizde root düğüm varsa organizasyonel şemanızı oluşturmaya başlayabilirsiniz demektir.Ahmet'in(root'un),altında çalışanları tanımlayabilmek için *GetDescendants* fonksiyonunu kullanacağız.Bu fonksiyon ana bir düğümden türeyen düğümleri,child nodeları, döndürmektedir.

Bu örneğimiz için ilk düğümü döndürecektir.

```
--Öncelikle dügümlerde tutulacak gecici degiskenler tanımlayalım.
DECLARE @YoneticiDugum hierarchyid
DECLARE @Sira hierarchyid

--Kok dugum,"Ahmet", icin ,Calisan_ID 5000 dir.
SELECT @YoneticiDugum=Dugum FROM Calisanlar WHERE Calisan_ID=5000
--GetDescendant ilk dugumu bize dondurecektir.Ilk dugum "/1"
INSERT INTO Calisanlar VALUES (@YoneticiDugum.GetDescendant(NULL,
NULL),5001, 'Ali')
```

Bu örneğimizde kök düğüm,Ahmet, basamak 0 dır ve "/ " ile temsil edilir.Aynı şekilde Ahmet'in altında çalışan Ali;basamak 1 dir ve Ahmet'den sonra en kıdemli kişidir, "/1" ile temsil edilir.Aynı şekilde Erdem'i de yapımıza eklediğimizde değeri "/2" olacaktır.

```
--Kok dugum,"Ahmet", icin ,Calisan_ID 5000 dir.
SELECT @YoneticiDugum=Dugum FROM Calisanlar WHERE Calisan_ID=5000
--GetDescendant ilk dugumu bize dondurecektir.Ilk dugum "/1"
INSERT INTO Calisanlar VALUES (@YoneticiDugum.GetDescendant(NULL,
NULL),5001, 'Ali')

--Ali'nin Calisan_ID di 5001 olacaktır.
SELECT @Sira=Dugum FROM Calisanlar WHERE Calisan_ID=5001
--Biz burada GetDescendant fonksiyonunu (Ali,NULL)olarak tanımlamış
oluyoruz.
--Eğer tam tersi bir şekilde (NULL,Ali) olarak tanımlasaydık,
--Ali'den sonra düğümler ekleyemeyecektik.

INSERT INTO Calisanlar VALUES
(@YoneticiDugum.GetDescendant(@Sira, NULL),5002, 'Erdem')
```

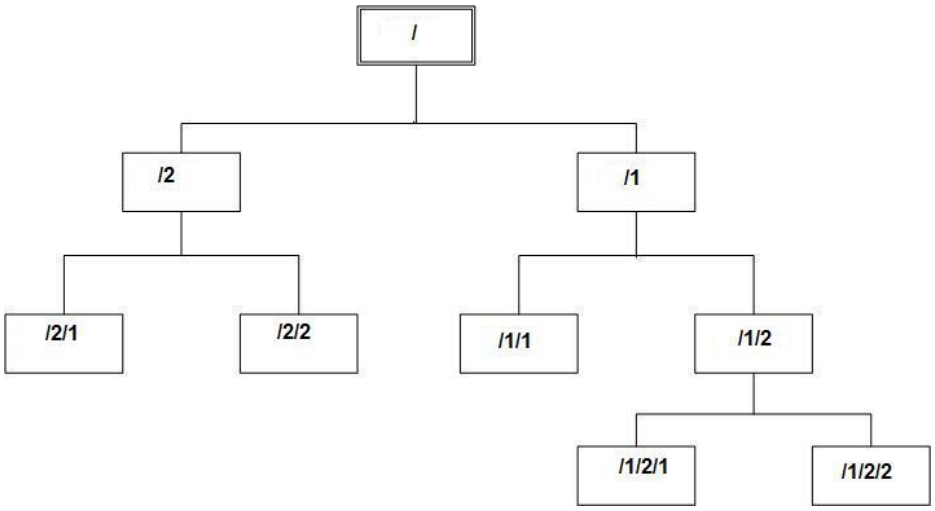
İsterseniz,şu ana kadarki oluşturduğumuz şirketimizin, hiyerarşik yapısını sorgulayalım

```
SELECT Dugum.ToString() AS Dugumun_Text_Gosterimi,
Dugum AS Dugumun_Binary_Gosterimi,
Dugum.GetLevel() AS Sira,
Calisan_ID,
Calisan_Isim
FROM Calisanlar
```

Results | Messages

	Dugumun_Text_Gosterimi	Dugumun_Binary_Gosterimi	Sira	Calisan_ID	Calisan_Isim
1	/	0x	0	5000	Ahmet
2	/1/	0x58	1	5001	Ali
3	/2/	0x68	1	5002	Erdem

Şirketimizin Organizasyon Şemasındaki,Dugumlerin yerleşimi ise şöyle olacaktır.



Şekilde de görüldüğü gibi Bilgehan,Erdem'in yöneticiliğinde çalışmaktadır ve düğümsel değeri "/2/1" dir.

SQL kodlarımızla şirketimizin organizasyon şemasını tamamlayalım.

```
SELECT @Sira=Dugum FROM Calisanlar WHERE Calisan_ID=5002
INSERT INTO Calisanlar VALUES (@Sira.GetDescendant(NULL, NULL),5003,
'Bilgehan')
--Erdem, yönetici olarak @Sira değişkeninde saklanıyor.
-- Bilgehan'ı ,Erdem altında çalışan biri olarak tanımladık.
DECLARE @dugum1 hierarchyid
--Bilgehan'ın,Calisan_ID si 5003 dür.
SELECT @dugum1=Dugum FROM Calisanlar WHERE Calisan_ID=5003
--GetDescendant,Bilgehan'dan sonraki Dugum 'ü döndürecek.
INSERT INTO Calisanlar VALUES (@Sira.GetDescendant(@dugum1, NULL),5004,
'Aydin')

--Şimdi de Aynı işlemleri Ali'nin Düğümü icin @Sira değişkeniyle
gerçekleştiriyoruz.
SELECT @Sira=Dugum FROM Calisanlar WHERE Calisan_ID=5001
--Ali'den türeyen bir düğüm yoktu(NULL,NULL).Şimdi Ali'den sonra Selen
düğümünü oluştuyoruz.
INSERT INTO Calisanlar VALUES (@Sira.GetDescendant(NULL,NULL),5005,
'Selen')
--Selen'in Dugum 'ü oluşturuldu.
SELECT @dugum1=Dugum FROM Calisanlar WHERE Calisan_ID=5005

--Tomris'i de Ali'nin yöneticiliğine atıyoruz.
INSERT INTO Calisanlar VALUES (@Sira.GetDescendant(@dugum1,NULL),5006,
'Tomris')
--Ada' da Tomris'in yöneticiliğinde
SELECT @Sira=Dugum FROM Calisanlar WHERE Calisan_ID=5006
--Ada'yı da Tomris'den sonra ki ilk yetkili yapıyoruz.
```

```
INSERT INTO Calisanlar VALUES (@Sira.GetDescendant(NULL,NULL),5007, 'Ada')
--Ada'nın Dugum 'ü olusturuldu.
SELECT @dugum1=Dugum FROM Calisanlar WHERE Calisan_ID=5007
--Cemil'i de ,Tomris'in yöneticiliğine sokuyoruz.
INSERT INTO Calisanlar VALUES (@Sira.GetDescendant(@dugum1,NULL),5008,
'Cemil')
```

Organizasyon şemamızı sorguladığımızda yapımız şöyle olmalıdır.

Results | Messages

	Dugumun_Text_Gosterimi	Dugumun_Binary_Gosterimi	Sira	Calisan_ID	Calisan_Isim
1	/	0x	0	5000	Ahmet
2	/1/	0x58	1	5001	Ali
3	/1/1/	0x5AC0	2	5005	Selen
4	/1/2/	0x5B40	2	5006	Tomris
5	/1/2/1/	0x5B56	3	5007	Ada
6	/1/2/2/	0x5B5A	3	5008	Cemil
7	/2/	0x68	1	5002	Erdem
8	/2/1/	0x6AC0	2	5003	Bilgehan
9	/2/2/	0x6B40	2	5004	Aydin

Diyelim ki organizasyonel yapımızda bir değişikliğe gidilmek zorunda kalındı ve Tomris ,Ali'nin ekibinden ayrılıp Aydın'ın ekibine dahil oldu.Böyle bir durum da ne olacak peki ? Tüm bu organizasyonel şema baştan mı oluşturulacak ? Tabii ki Hayır :=)

İşte bu tür;hiyerarşik yapıdaki değişiklikler için, **GetReparentedValue()** fonksiyonu kullanılmalıdır. (SQL Server 2008 RC0 'dan itibaren Reparent() fonksiyonu,**GetReparentedValue() olarak değiştirilmiştir)**

```
DECLARE @Basamak hierarchyid
DECLARE @EskiYonetici hierarchyid
DECLARE @YeniYonetici hierarchyid
SELECT @Basamak=Dugum from Calisanlar where Calisan_ID=5006 -- Tomris
SELECT @EskiYonetici=Dugum from Calisanlar where Calisan_ID=5001 -- Ali
artık Tomris'i yönetmeyecek.
SELECT @YeniYonetici=Dugum from Calisanlar where Calisan_ID =5004 -- Aydın
artık Tomris'in yöneticisi
UPDATE Calisanlar
SET Dugum = @Basamak.GetReparentedValue(@EskiYonetici, @YeniYonetici)
WHERE Dugum = @Basamak
```

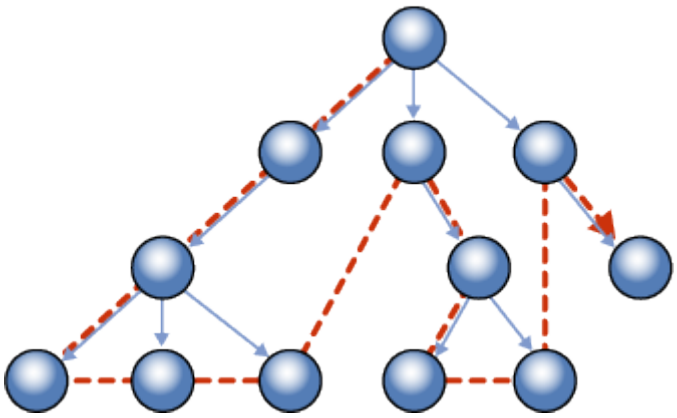
Tablomuzun son haline bakarsak,aşağıdaki gibi değiştiğini göreceksiniz.

	Dugumun_Text_Gosterimi	Dugumun_Binary_Gosterimi	Sira	Calisan_ID	Calisan_Isim
1	/	0x	0	5000	Ahmet
2	/1/	0x58	1	5001	Ali
3	/1/1/	0x5AC0	2	5005	Selen
4	/1/2/1/	0x5B56	3	5007	Ada
5	/1/2/2/	0x5B5A	3	5008	Cemil
6	/2/	0x68	1	5002	Erdem
7	/2/1/	0x6AC0	2	5003	Bilgehan
8	/2/2/	0x6B40	2	5004	Aydin
9	/2/2/2/	0x6B5A	3	5006	Tomris

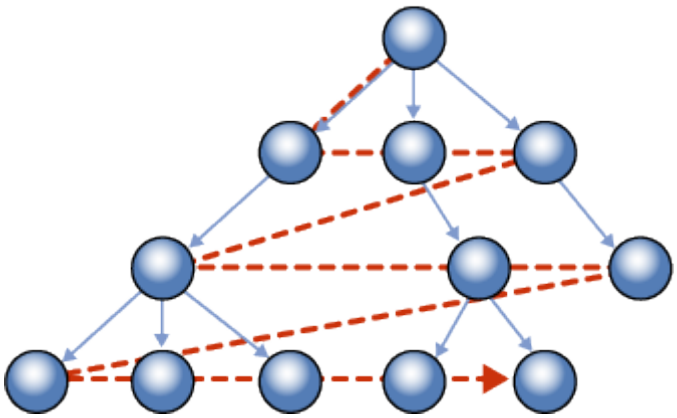
Görüldüğü gibi;Tomris artık Aydın tarafından yönetilen bir çalışan.

Burada dikkatinizi çektiyse,küçük bir problemle karşı karşıyayız.Şöyle ki Tomris artık Ali tarafından yönetilmekten çıkıp Aydın tarafından yönetilmekte fakat Tomris'in yönettiği çalışanlar yöneticisiz kalmış durumda. ("/1/2" şeklinde bir düğüm artık bulunmamakta)

Böyle bir durum sıkıntı yaratacaksa,geliştiriciler olarak duruma müdahale edilmeli ve indexler oluşturulmalıdır.Bu örnek için, 2 tür index tanımlaması yapılabilir.İlki Depth-First Index'dir ve bu index yapısında bütün düğümler,yanyana dizilir,birlikte saklanır ve istenilen özel bir düğüme bağlanabilir.Bütün çalışanların sadece bir yöneticiye bağlı olarak çalıştığı yapılarda kullanılabilir.



İkinci index türümüz ise;Breadth-First Index'dir.Bu index türünde bütün düğümler,dereceleriyle birlikte sıralı olarak tutulur.Aynı yöneticiye bağlı çalışanlar birlikte tutulurlar.



Bir tablo oluşturulduktan sonra,depth first index,direk olarak primary key ile tanımlanmış olur.(Yani default olarak depth first index tanımlanmıştır.)Breadth-First Index tanımlamasını ise şöyle bir komutla biz yapabiliriz.

```
CREATE Index Breadth on Calisanlar(Dugum,Basamak)
```

Bu makalemizde SQL Server 2008 'in yeni veri tiplerinden olan Hierarchyid *Veri Tiplerini ve kullandığı fonksiyonlarını,örnek sorgularla inceledik.Bu makaleye bağlı makalelerde SQL Server 2008 ile gelen yeni veri tiplerini incelemeye devam edeceğiz.*

Başka bir Microsoft SQL Server 2008 makalesinde görüşmek dileğiyle...

Bilgehan GÜRÜNLÜ

www.gurunlu.com
bilgehan@gurunlu.com

Kaynaklar :

-http://msdn.microsoft.com/en-us/library/
-Accelerated SQL Server 2008
-Programming Microsoft® SQL Server 2008 (Ms-Press)
- SQL Server TechCenter