Programlama Performans İpucu: (&&) ve (||) Operatörleri Kullanılırken

Operand (x)	operand (y)	Value of the Expression		
		x || y	x && y	x ! y
true	true	true	true	false
true	false	true	false	true
false	true	true	false	true
false	false	false	false	true

1. VE (&&) operatörünün doğru sonucu dönmesi için bütün şartlarının doğru olması gerekmektedir. Bu sebeple && operatörü kullanılırken karşılaşılan ilk yanlış şartta geriye kalan şartlar sorgulanmadan program sonraki satıra geçecektir. && operatörünün bu özelliğinden dolayı yanlış olma ihtimali yüksek olan şartlar başta yazılırsa performans artacaktır.
2. VEYA (||) operatörünün doğru sonucu dönmesi için herhangi bir şartın doğru olması yetmektedir. Bu sebeple || operatörü kullanılırken karşılaşılan ilk doğru şartta geriye kalan şartlar sorgulanmadan program sonraki satıra geçecektir. || operatörünün bu özelliğinden dolayı doğru olma ihtimali yüksek olan şartlar başta yazılırsa performans artacaktır.

Enum Değerlerini foreach Dönmek

Enum değerleri yazılım geliştiricilerin en sık kullandığı ve en faydalı yapılardan biridir. Bazı durumlarda enum yapısının içerdiği bütün değerleri dolaşmak gerekmektedir. Bu durumda kullanılan iki yöntem bulunmaktadır.

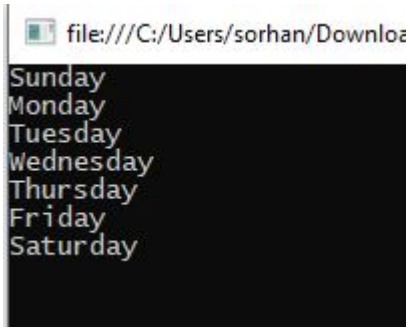
Enum.GetNames

Parametre olarak bir enum tipi alır. Sonuç olarak aldığı enum tipinin barındığı değerlerin adını içeren bir string dizisi döner.

Aşağıdaki örnek kodda hafta günlerini barındıran DayOfWeek enum yapısı foreach dönülmektedir.

```
string[] weekdays = Enum.GetNames(typeof(DayOfWeek));

            foreach (string weekday in weekdays)
            {
                Console.WriteLine(weekday);
            }
```



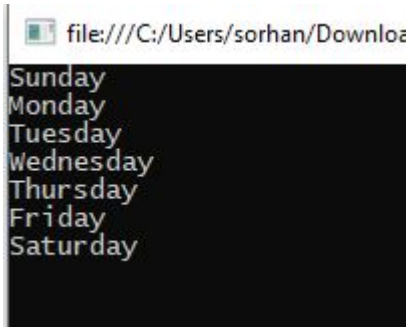
Enum.GetValues

Bu metot da GetNames metodu gibi bir enum tipi alır. Dönüş değeri olarak da içerisinde aldığı enum tipinin değerlerini barındıran bir dizi (Array) dönderir. Bu Array değerinin elemanları enum'un türüne dönüştürülerek kullanılabilir.

Aşağıdaki kodda örnek bir foreach bulunmaktadır.

```
                                    Array   weekdays   =
Enum.GetValues(typeof(DayOfWeek));

            foreach (DayOfWeek weekday in weekdays)
            {
                Console.WriteLine(weekday);
            }
```



Not: typeof metodu parametre olarak adığı tür adının Type türünden tipini dönderir.

Git Sistemini Güncelleme

Git, versiyon kontrol sistemleri içerisinde en yaygın kullanılan olanıdır. Standart bir arayüz ile kullanılmadığından çoğunlukla güncellemesi ihmal edilen bir sistem oluyor. Aşağıdaki adımlar takip edilerek lokalde kurulu olan git sistemi güncellenebilir.

```
git --version
```

kodu ile mevcutta kullanılan git versiyonu tespit edilir.

- Mevcutta kullanılan versiyon 2.14.1 versiyonundan eski bir versiyon ise mevcut kurulum tamamen kaldırılıp yeni versiyon yüklenir.
- Mevcutta kullanılan versiyon 2.14.2 ve 2.16.1 versiyonları arasında ise aşağıdaki kod çalıştırılarak güncellenir.

```
git update
```

- Mevcutta kullanılan versiyon 2.16.2 versiyonu ve sonrası ise aşağıdaki kod çalıştırılarak git sistemi güncellenir.

```
git update-git-for-windows
```

Android Butona Click Event Atama

C# Extension Metotları Oluşturma

C# İle SQL Server İşlemleri

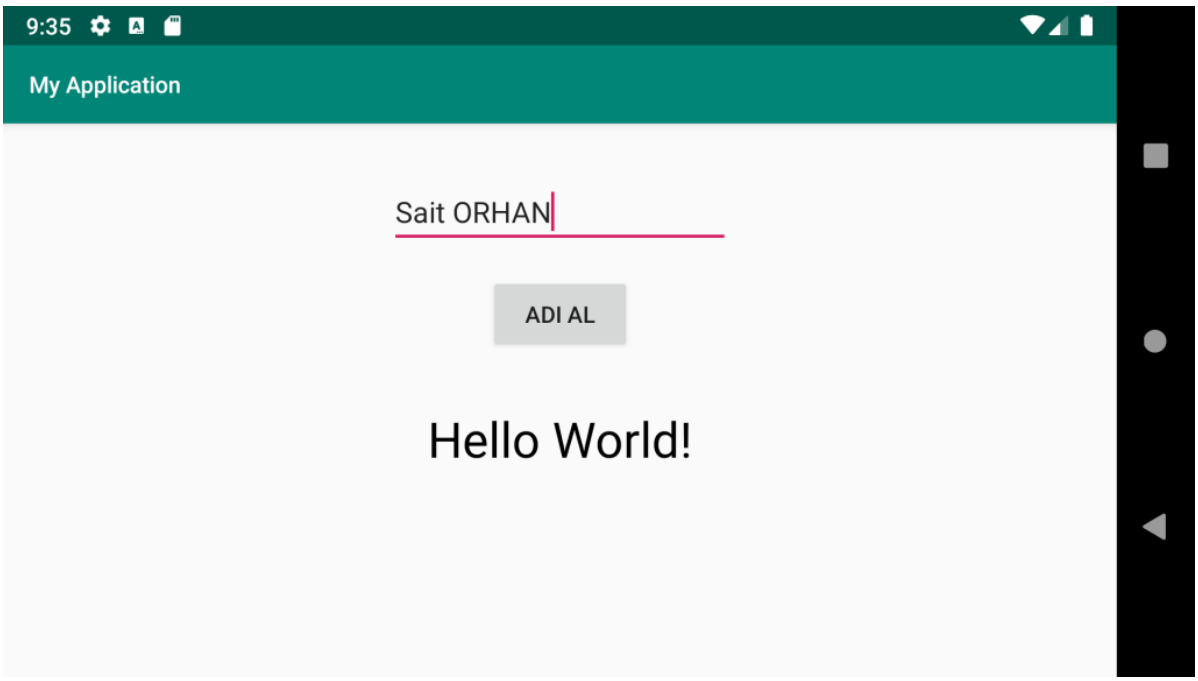
Yazılım geliştirme çoğunlukla veri saklama ve işleme amacıyla yapılır. Verilerin saklanması içinde bir veritabannı sistemi kullanılır. Videomuzda en çok kullanılan veritabanı sistemlerinden olan SQL Server'a C# ile işlem yapmayı öğreneceğiz.

Instance State Nedir, Ne Zaman Kullanılır?

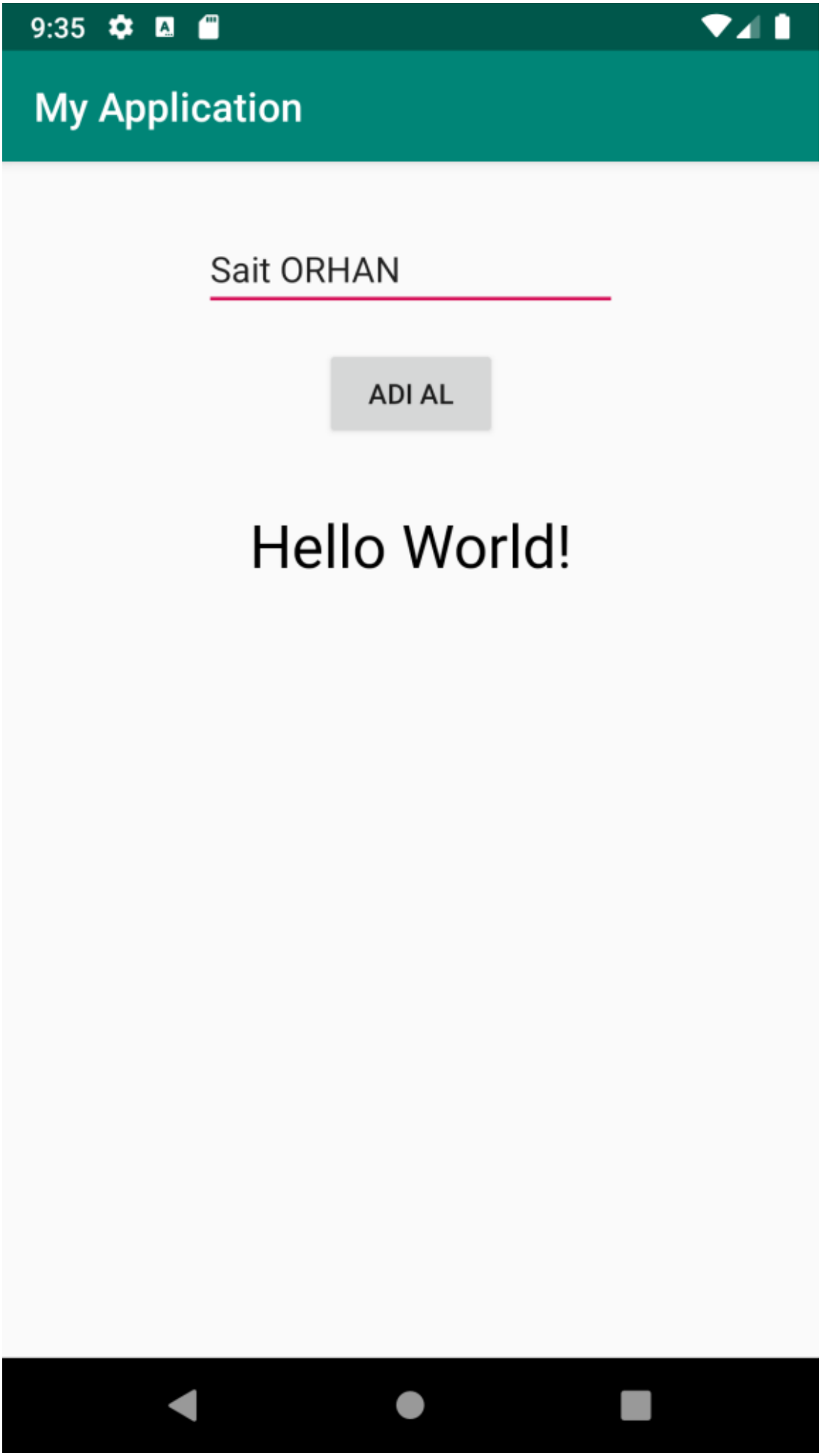
Konunun içeriğinin daha iyi anlaşılması açısından şöyle bir senaryo üzerinden ilerleyebiliriz:


9:35
My Application
Sait ORHAN
ADI AL
Sait ORHAN

Ekranın Orijinal Durumu



Ekranın Yataya Çevrildikten Sonraki Durumu


9:35
My Application
Sait ORHAN
ADI AL
Hello World!

Ekranın Tekrar Orijinal Yönüne Döndürülmüş Hali

Örnek ekranımızda bir adet TEXTBOX ve bir adet LABEL var. Butona tıklandığında TEXTBOX değerini alıp LABELe yazdırılıyor. İlk ekranımızda göründüğü gibi “Sait ORHAN” yazısı labele yazdırılmış. Bu durumda iken telefonun yönünü değiştirilmesi durumunda label değeri tasarım aşamasında verilen değere dönüyor. Örneğimizde “HELLO WORLD!” metnine döndüğü gibi ve telefon yönü eski durumuna getirilse bile atanan değer geri gelmiyor.

Bu durumun nedeni Android‘in kullanıcıdan alınan değerleri tutup performansa olumsuz etkisi olmaması için statik değer olduğu farzedilen değerleri hafızaya almamasıdır.

Bu sorunun çözmenin yolu; telefonun yönü değiştiğinde sırası ile onSaveInstanceState ve onRestoreInstanceState olayları tetikleniyor.

onSaveInstanceState metodu içerisinde bu şekilde kaybolan değerler Bundle tipindeki parametreye alınarak koruma altına alınır.

```
@Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);

                                outState.putString("name",
labelName.getText().toString());
    }
```

onRestoreInstanceState metodu içinde de onSaveInstanceState

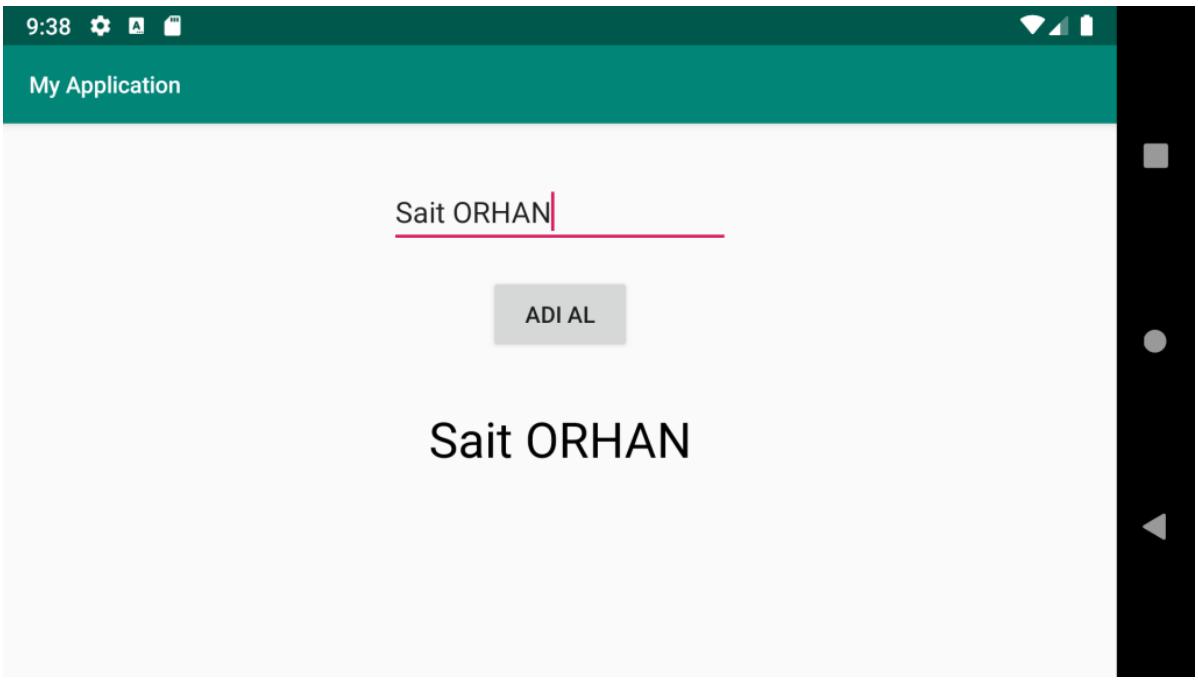
metodunda kaydedilen değerler alınarak ilgili kontrollere atanır.

```
@Override
        protected void onRestoreInstanceState(Bundle
savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);

        String name = savedInstanceState.getString("name");
        labelName.setText(name);
    }
```

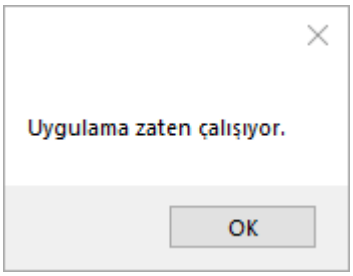

9:35
My Application
Sait ORHAN
ADI AL
Sait ORHAN

Ekranın Orijinal Durumu



İşlemlerden Sonra Telefonun Yatay Çevrilmiş Durumu

Uygulamanın Tek Örneğinin Çalışmasına Müsaade Etmek



Bazı uygulamalar çalışma prensipleri gereği aynı anda tek örneğinin çalışmasına müsaade eder. Uygulama açık iken başka bir örneği çalıştırılmak istendiğinde yukarıdaki gibi uyarı

mesajı verip açılacak olan son örneği iptal eder.

Bu işlem için **System.Diagnostics** sınıfına ait Process sınıfından yararlanılır. Aşağıdaki kod örneğinde görüldüğü gibi öncelikle **Process.GetCurrentProcess().ProcessName** kod parçası ile başlatılan uygulamanın işlem adı alınıyor. **Process.GetProcessesByName** metodu ile de bu isme sahip işlemlerin listesi alınıyor. Eğer bu isimde birden fazla işlem var ise uyarı mesajı verilip return komutu ile işlem iptal edilir ve uygulama sonlandırılır.

```
                        Process[] processesByName = Process.GetProcessesByName(Process.GetCurrentProcess().ProcessName);
            if (processesByName.Length > 1)
            {
                MessageBox.Show("Uygulama zaten çalışıyor.");
                return;
            }
```

Kodu uygulamanızın Main metodunun içinde başlangıç formunu ekrana göstermeden önce kullanabilirsiniz. Bu şekilde kullanmanız durumunda Main metodunuz aşağıdaki gibi olacaktır.

```
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);

                        Process[] processesByName = Process.GetProcessesByName(Process.GetCurrentProcess().ProcessName);
            if (processesByName.Length > 1)
            {
```

```
                MessageBox.Show("Uygulama zaten çalışıyor.");
                return;
            }

            Application.Run(new Form1());
        }
```

C# String Değerden Diğer Veri Türlerine Çeviri

C# dilinin klavyeden girilen varsayılan veri tipi string türüdür. Alınan verinin kullanım yerine göre özellikle sayı türüne çevrilmesi gerekmektedir. Bu işlem yapılırken temel olarak üç yöntemden biri kullanılabilir. Bu yöntemler aşağıdaki gibidir:

- Convert sınıfının ilgili metotlarından birini kullanmak
- Hedef sınıfın Parse adlı metodunu kullanmak
- Hedef sınıfın TryParse adlı metodunu kullanmak

Convert sınıfının ilgili metotları “To” ile başlayan ve hedef türün adı ile biten metotlardır. ToInt16, ToInt32, ToDouble vb…

İlk örneğimizi Convert sınıfı ile verelim:

```
string metin = textBoxNumber.Text;
if (metin.Any(c => !Char.IsDigit(c)))
{
    MessageBox.Show("Hatalı giriş");
    return;
}
int sayi = Convert.ToInt32(metin);
```

Bu örneğimizde if şartının içinde yer alan şartta girilen metnin sadece rakamlardan oluşup oluşmadığı test ediliyor. Rakam dışında bir karakter barındıran bir metin geldi ise “Hatalı giriş” uyarısını verip işlem return oluyor.

Aynı işlemi Parse metodu ile yapacak olursak kodu aşağıdaki gibi değiştirmemiz gerekmektedir.

```
string metin = textBoxNumber.Text;

if (metin.Any(c => !Char.IsDigit(c)))
{
    MessageBox.Show("Hatalı giriş");
    return;
}

int sayi = Int32.Parse(metin);
```

Parse metodunun kullanımı da Convert sınıfının ToXXX metotlarının kullanımı ile aynıdır.

string to int çeviri işleminde kullanılan bir diğer metot da TryParse metodudur. Bu metodun kullanımı sırasında gelen metnin sadece rakamlardan oluşup oluşmadığını test etmeye gerek yoktur. Çalışma şeklini örnek kod üzerinden inceleyelim:

```
string metin = textBoxNumber.Text;
int sayi;

bool tryParse = Int32.TryParse(metin, out sayi);
if (tryParse)
{
    MessageBox.Show("Doğru format");
}
else
{
    MessageBox.Show("Yanlış format");
}
```

TryParse metodunun geri dönüş tipi diğerlerinin aksine bool veri tipidir. Çalışma şekli de aşağıdaki adımlardan oluşmaktadır.

1. Hedef veri türünde bir tanımla yapılır. (Satır 2)
2. İlgili türün TryParse metodu çağrılır. İlk parametreye dönüştürülecek kaynak metin, ikinci parametreye de daha önce oluşturulan hedef türün örneği out parametresi ile verilir. (Satır 4)
3. Bu metot çağırma sonucunda dönüşüm işlemi başarılı olursa hedef türün örneğine dönüşüm sonucu atanır, metot dönüş değeri de true olur. Dönüşümün başarısız olması durumunda ise hedef türün örneğine ilgili sınıfın varsayılan değeri atanır, metot dönüş değeri de false olur. Böylelikle işlemin sonucu bir if yardımı ile sorgulanmış olur.

Nesnelerin SQL Server Üzerinde Kullandığı RAM Miktarını Bulma

SQL Server üzerinde yer alan veritabanı ve diğer nesnelerin ne kadar sistem kaynağı tükettiğini bulmak için aşağıdaki iki temel sorguyu kullanabiliriz.

İlk sorgumuzda veritablanı bazında kullanılan RAM miktarını sorgulayabiliriz.

```
SELECT
[DatabaseName] = CASE [database_id] WHEN 32767
THEN 'Resource DB'
ELSE DB_NAME([database_id]) END,
COUNT_BIG(*) [Pages in Buffer],
COUNT_BIG(*)/128 [Buffer Size in MB]
FROM sys.dm_os_buffer_descriptors
GROUP BY [database_id]
ORDER BY [Pages in Buffer] DESC;
```

	DatabaseName	Pages in Buffer	Buffer Size in MB
1	Tfs_Configuration	19203	150
2	HascelikDocs	15240	119
3	Tfs_MetalMatris	6575	51
4	MasterSecurity9	5113	39
5	tempdb	3188	24
6	Resource DB	2781	21
7	HascelikDof	962	7
8	msdb	564	4
9	HasCelikMisafirDB	306	2
10	master	256	2
11	YYS	213	1
12	MMCustumerFeed	182	1
13	MmPriceTrack	179	1
14	hascelikcertification	175	1
15	Inventory	173	1
16	HascelikMetaDatas	169	1
17	PTS_Metcom	167	1
18	model	29	0

Veritabanı Kullanılan RAM Sorgusu

Diğer sorgumuzda ise Index ve benzeri diğer nesnelerin kullandığı RAM miktarını sorgulayabiliriz.

```
SELECT obj.name [Object Name], o.type_desc [Object Type],
i.name [Index Name], i.type_desc [Index Type],
COUNT(*) AS [Cached Pages Count],
COUNT(*)/128 AS [Cached Pages In MB]
FROM sys.dm_os_buffer_descriptors AS bd
INNER JOIN
(
SELECT object_name(object_id) AS name, object_id
,index_id ,allocation_unit_id
FROM sys.allocation_units AS au
INNER JOIN sys.partitions AS p
ON au.container_id = p.hobt_id
AND (au.type = 1 OR au.type = 3)
UNION ALL
SELECT object_name(object_id) AS name, object_id
,index_id, allocation_unit_id
FROM sys.allocation_units AS au
INNER JOIN sys.partitions AS p
```

```
ON au.container_id = p.partition_id
AND au.type = 2
) AS obj
ON bd.allocation_unit_id = obj.allocation_unit_id
INNER JOIN sys.indexes i ON obj.[object_id] = i.[object_id]
INNER JOIN sys.objects o ON obj.[object_id] = o.[object_id]
WHERE database_id = DB_ID()
GROUP BY obj.name, i.type_desc, o.type_desc,i.name
ORDER BY [Cached Pages In MB] DESC;
```

Results | Messages

	Object Name	Object Type	Index Name	Index Type	Cached Pages Count	Cached Pages In MB
1	syssingleobjrefs	SYSTEM_TABLE	clst	CLUSTERED	4	0
2	sysdbreg	SYSTEM_TABLE	nc2	NONCLUSTERED	3	0
3	syscerts	SYSTEM_TABLE	nc2	NONCLUSTERED	4	0
4	sysiscols	SYSTEM_TABLE	clst	CLUSTERED	4	0
5	sysasymkeys	SYSTEM_TABLE	nc3	NONCLUSTERED	1	0
6	sysallocunits	SYSTEM_TABLE	nc	NONCLUSTERED	5	0
7	syspru	SYSTEM_TABLE	cl	CLUSTERED	2	0
8	sysbrickfiles	SYSTEM_TABLE	clst	CLUSTERED	5	0
9	syscerts	SYSTEM_TABLE	nc1	NONCLUSTERED	4	0
10	sysxlgns	SYSTEM_TABLE	nc2	NONCLUSTERED	3	0
11	syscerts	SYSTEM_TABLE	cl	CLUSTERED	4	0
12	sysdbreg	SYSTEM_TABLE	nc1	NONCLUSTERED	3	0
13	sysowners	SYSTEM_TABLE	clst	CLUSTERED	3	0
14	sysxlgns	SYSTEM_TABLE	cl	CLUSTERED	3	0
15	sysnsobjs	SYSTEM_TABLE	clst	CLUSTERED	1	0
16	syscolpars	SYSTEM_TABLE	nc	NONCLUSTERED	32	0
17	syssingleobjrefs	SYSTEM_TABLE	nc1	NONCLUSTERED	4	0
18	syscolpars	SYSTEM_TABLE	clst	CLUSTERED	32	0
19	sysnsobjs	SYSTEM_TABLE	nc	NONCLUSTERED	1	0
20	sysscalartypes	SYSTEM_TABLE	nc1	NONCLUSTERED	3	0
21	sysxlgns	SYSTEM_TABLE	nc1	NONCLUSTERED	3	0
22	sysrscols	SYSTEM_TABLE	clst	CLUSTERED	13	0
23	sysschobjs	SYSTEM_TABLE	nc1	NONCLUSTERED	70	0
24	sysidxstats	SYSTEM_TABLE	clst	CLUSTERED	5	0
25	syslnklgns	SYSTEM_TABLE	cl	CLUSTERED	1	0
26	sysasymkeys	SYSTEM_TABLE	cl	CLUSTERED	1	0
27	sysallocunits	SYSTEM_TABLE	clust	CLUSTERED	5	0
28	sysschobjs	SYSTEM_TABLE	clst	CLUSTERED	70	0

Nesnelerin Kullandığı RAM Miktarı