T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

# MS SQL SERVER 2000 VERİTABANI'NDA PERFORMANS DENETİMİ VE OPTİMİZASYONU

**YÜKSEK LİSANS TEZİ**

**Bilg.Müh. Osman AYHAN**

**Enstitü Anabilim Dalı : BİLG. VE BİLİŞİM. MÜH.**

**Tez Danışmanı : Yrd. Doç. Dr. Fevzullah Temurtaş**

**Aralık 2006**

**T.C.**
**SAKARYA ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# MS SQL SERVER 2000 VERİTABANI'NDA PERFORMANS DENETİMİ VE OPTİMİZASYONU

**YÜKSEK LİSANS TEZİ**

**Bilg. Müh. Osman AYHAN**

**Enstitü Anabilim Dalı : BİLG. VE BİLİŞİM. MÜH.**

**Bu tez 01 / 02 /2007 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.**

| Prof. Dr. Ümit Kocabıçak | Prof. Dr. Osman Çerezci | Doç. Dr. Fevzullah Temurtaş |
|---|---|---|
| **Jüri Başkanı** | **Üye** | **Üye** |

# TEŞEKKÜR

Bu tezin hazırlanması sürecinde hoşgörüsünü ve yardımlarını esirgemeyen danışmanım Sayın Doç. Dr. Fevzullah Temurtaş'a, yüksek lisans sürecinde, okula devam edebilmem için vermiş olduğu izinlerden dolayı ve çalışmamı canlı bir ortamda test etme olanağını bana sunan Sandoz İlaç Bilgi Sistemleri Direktörü Sayın N. Yekta Caymaz'a, tez çalışmam boyunca çevirileri yapmamda büyük yardımı olan arkadaşım Gürcan Yılmaz'a teşekkürlerimi bir borç bilirim.

Tüm hayatım boyunca ve yüksek lisans sürecinde benden desteklerini ve yardımlarını ve sevgilerini esirgemeyen çok sevgili ailem ve arkadaşlarıma da ayrıca teşekkür ederim.

# İÇİNDEKİLER

## SİMGELER VE KISALTMALAR LİSTESİ

| | |
|---|---|
| SQL | :Structured Query Language |
| T-sql | :Transact Structured Query Language |
| ADO | :ActiveX Data Objects |
| OLED DB | : Object Linking and Embedding Database |
| ODBC | : Open Database Connectivity |
| MS SQL | :Microsoft Structured Query Language |
| DSN | :Data Source Name |
| API | :Application Programming Interface |
| SLA | :Service Level Agreement |
| OLAP | : Online Analytical Processing |
| OLTP | :Online Transaction Processing |
| DTS | :Data Transfer Service |

# ŞEKİLLER LİSTESİ

# TABLOLAR LİSTESİ

# ÖZET

Anahtar kelimeler: Veritabanı performansı, veritabanı performans optimizasyonu, SQL Server 2000

Sahip olunan veri miktarı arttıkça bu verilerin doğru ve hızlı bir şekilde depolanması ve kullanılması konusunun önemi de artmaktadır. Yüksek performansa sahip bir uygulama veritabanı bütünlüğünden söz edebilmek için, sahip olunan uygulamalar ne kadar performansı yüksek uygulamalar olursa olsun, bu uygulamaları besleyen veritabanı sunucularının performansı da bunlara ayak uydurabilecek düzeyde olması gereklidir. Veritabanı sistemleri de iç ve dış olmak üzere bir çok etkenden etkilenerek performanslarında değişiklik gösterebilmektedirler.

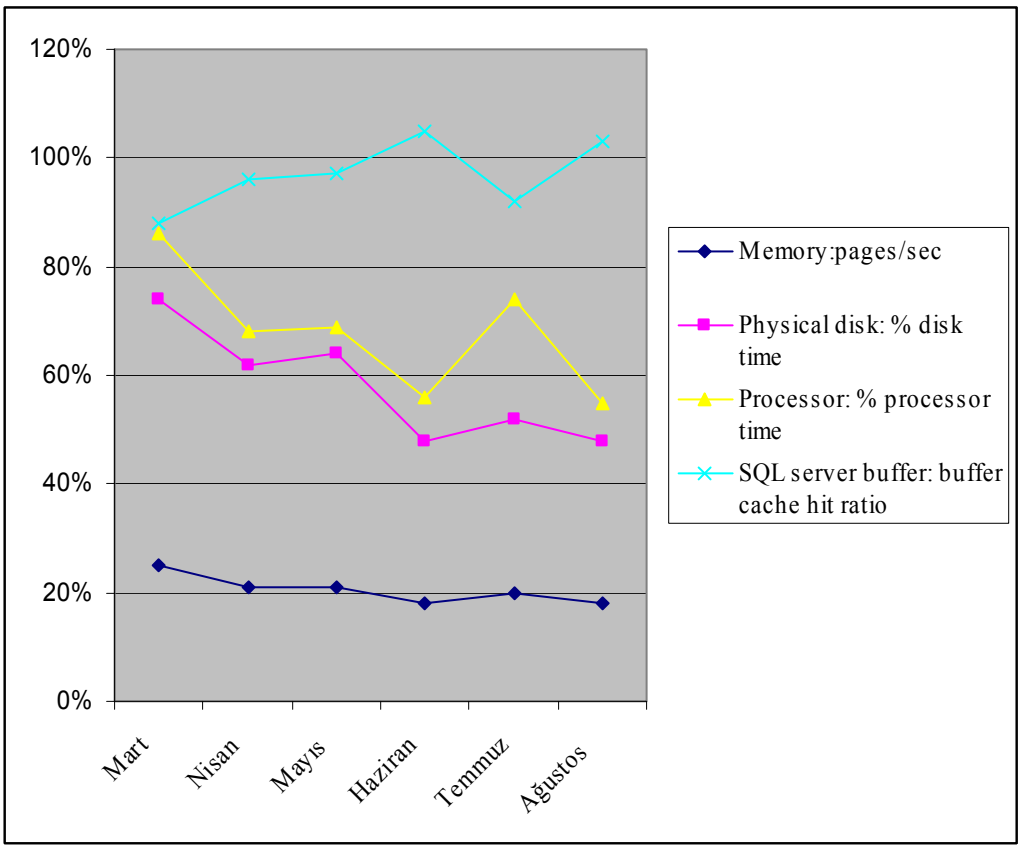
MS SQL Server 2000 veritabanı da, donanımsal etkenler, üzerinde koştuğu işletim sisteminin etkileri, veritabanı konfigürasyonu ayarları , sunucu konfigürasyonu ayarları, indeks ayarları, sorgu düzenlemeleri vb. gibi konulardan etkilenerek, bu sunucuyu kullanan uygulamalara, değişken cevap süreleri sunmaktadır. Sistemin çalışma performansını etkileyen bu kadar çok etken varken de bütün performans düzenleme ve denetleme çalışmalarının bir sistematik içerisinde yapma gereksinimi doğmaktadır. Bütün bu etkenlerin madde madde incelenmesi halinde sunucu performansına etki eden bu maddelerin bir çok alt dalı da olduğu görünür. Bütün bu alt ayarlar yapıldığı zamanda sistemin toplam performansı ortaya çıkar. Bu alt etkenlerin sistem üzerinde tek tek etkisi var iken bir de hep beraber uygulandıkları zaman sistem üzerinde olumlu veya olumsuz etkileri bulunmaktadır.

Bu araştırmada bütün bu iç ve dış etkenler incelenip 6 ay boyunca 400 kullanıcısı olan, veritabanı sunucusu olarak SQL Server 2000 kullanan bir saha otomasyonu projesinde uygulanmış ve optimum çözümler aranmıştır. Bu bağlamda, donanın darboğazları denetimi kontrol listesi, performans kontrol denetim listesi, işletim sistemi performans değerlendirme kontrol listesi, SQL Server konfigürasyonu performans kontrol listesi, SQL Server ayarları kontrol listesi, SQL Server indeks performansı kontrol listesi, SQL Server sorgu performansı kontrol listesi, uygulama kontrol listesi ve SQL Server görevler kontrol listesi oluşturulmuştur. Bu listelerde ki bütün maddeler tek tek incelenip her bir etkenin olması gereken değer ve bu değerlerin artan azalan değerlerinin bizlere neler ifade etmesi gerektiği ortaya konmuştur. Oluşturulan bu kontrol listeleri 6 ay boyunca saha otomasyonu projesinde uygulanmış ve sonuç bölümünde ortaya çıkan değişimler ortaya konulmuştur. İndekslerin, t-sql cümlelerinin ve depolanmış yordamların kullanım özelliklerine göre, sorgu performanslarında olan değişiklikler gözlenerek, en optimum t-sql cümlesini öneren, vb.net koduyla yazılmış olan “konteskuel” kod isimli bir program elde edilmiştir.

# BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Bu tez araştırması boyunca anlatılan bütün optimizasyon kuralları, standart önerilerden yola çıkılarak, yapılan uygulamalar sonucunda, Server'ın vermiş olduğu tepkiler gözlemlenerek bütün optimizasyon kuralları artısıyla eksisiyle ortaya koyulmuştur. Dış etkenlerden olan SQL Server donanım performansı konusu canlı ortamda test edilememiştir. Bu etkenin, yapılan değişiklikler öncesi ve sonrasındaki yarattığı farklılıkların gözlenebilmesi için gerekli olan operasyonun maddi maliyeti yüksek olduğu için, işlemci değiştirmek, RAM arttırmak, disk dizilerinin tipini değiştirmek gibi değişikliklere gidilememiş ancak yapılması gerekenler madde madde listelenmiştir.

Altı aylık dönemin, üç aylık ilk döneminde SQL Server 2000 bütün mevcut varsayılan değerleriyle konfigüre edilmiş ve performans görüntüleme aracıyla takip edilmiştir. İkinci 3 aylık periyotta ise çalışma boyunca bahsedilen ve önerilen değişiklikler sunucuya uygulanmıştır ve sunucunun hem kaynakları aşırı kullanımının önüne geçilmiş, hem de sorgulara verilen cevaplar hızlanmıştır. Uygulamanın ilk başlarında gelen yavaşlık şikayetlerinin önüne geçilmiştir. Çalışmanın aşamalarının test edildiği makinenin ilgili donanım konfigürasyonu şöyledir; HP Series DL380 G4, Xeon 3.2 Ghz Dual Core CPU, 4 GB RAM, 410 GB data disk, RAID 5 disk dizisi. Sistemin üzerinde koştuğu işletim sistemi ise MS Server 2003'dür. Bu sistem üzerindeki test edilen çalışmanın 6 aylık izleme sonucunda, sistem kaynaklarının kullanımı ve gelen sorgulara verdiği cevapların performansı aşağıdaki grafiklerlerde görülmektedir.




Şekil 2.4 Kaynakların kullanımının zamanla değişim grafiği (1)

Mart 2006'dan Mayıs 2006 sonuna kadar SQL Server 2000'in varsayılan değerleriyle çalışılmıştır. Bu sunucudan beslenen saha otomasyonu projesi, ilaç firmasının saha ekibinin günlük bütün işlemlerini üzerinde yaptığı bir uygulamadır. Mart ve temmuz aylarında, şirkette kampanya dönemleri olmuştur ve tıbbi satış mümessillerinin satış ve ziyaret aktiviteleri artmıştır. Bu önbilgiler eşliğinde grafik incelenecek olursa;

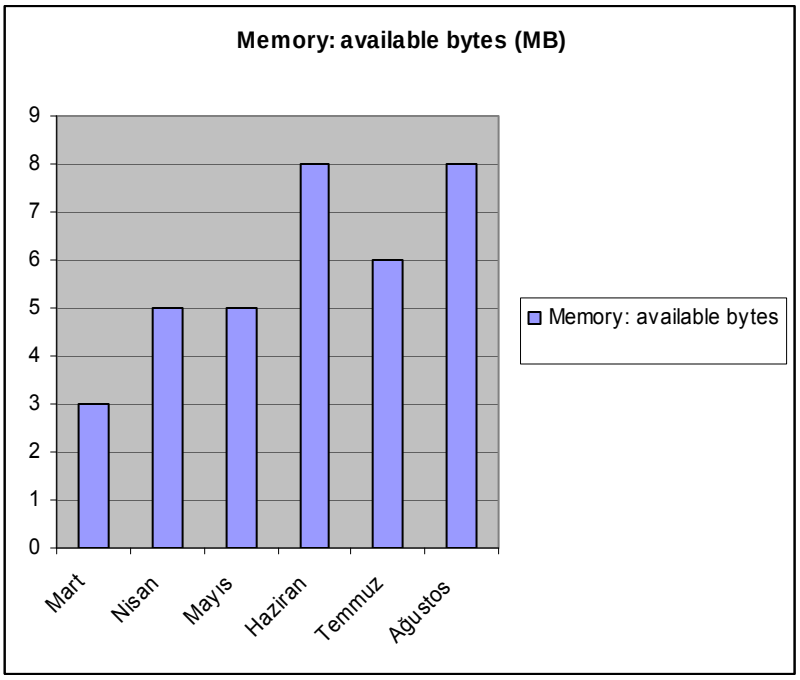
- Memory:pages/sec: Bu sayaç , RAM'den diske saniyede geçen sayfa sayısını belirtir. Ne kadar fazla sayfa geçişi olursa server üzerinde o kadar fazla I/O yükü biner. Bu da performansı olumsuz yönde etkiler. Bu sayacın yüzdesi 0 ile 20 arasında olmalıdır. Buradaki anahtar cümle ortalama sayfanın %20'den az olması gerekliliğidir. Bu oranın %20'den fazla olması RAM ihtiyacını ortaya çıkartır. Grafikte görüldüğü üzere mart-

nisan döneminde bu oran %25 seviyesindedir. Buradan da anlaşılacağı gibi, SQL Server'ın RAM ihtiyacı oldukça yüksek olmuş ve sistemde RAM takviyesi için gerekli veriler ortaya çıkmıştır. Kampanya döneminin olmadığı nisan ve mayıs aylarında bu oran %21'lerde ölçülmüştür. Sistem normal bir şekilde devam ederken bile sisteme RAM takviyesi gereklidir. Ancak bu çalışma boyunca anlatılan denetim çalışmaları sonucunda ortaya çıkan sonuçlar doğrultusunda, haziran, temmuz ve ağustos aylarında yapılan optimizasyon hareketlerinden sonra, daha ilk aydan bu performans ölçütünde %18 seviyesine düşüş gözlenmiştir. Sistem kendisi için gerekli olan RAM ihtiyacını daha verimli bir şekilde kullanmış ve RAM takviyesine olan ihtiyaç ortadan kalkmıştır. Ancak optimizasyon çalışmaları yapılmış olsa da yine bu dönemdeki kampanya çalışmalarından dolayı temmuz ayında bu oran %20 oranına yükselmiştir.

-Physical disk: % disk time: Bu sayacın ölçtüğü değer ise fiziksel bir diskin ne kadar meşgul olup olmadığıdır. Dizilerin ne kadar meşgul olduğunu görmek için güzel bir karşılaştırma verisi verir. Bir kural olarak % Disk Time sayacı %55'den daha küçük olmalıdır. Eğer bu yüzdenin üzerinde bir seviyede bulunuyorsa SQL Server'da bir darboğaz oluşacak denilebilir. Grafikten görüleceği üzere bu değer mart ayı için, kabul edilir değerlerin çok üzerinde ve %74 seviyesinde. Halbuki bir diğer kampanya dönemi olan ve yoğun olarak OLTP işlemlerinin yapıldığı temmuz ayında bu değer yalnızca %52 seviyesinde. Diğer ayların karşılaştırılması da yapıldığı zaman, gerekli optimizasyon çalışmalarının yapıldığı, haziran, temmuz, ağustos dönemlerinde değerlerin kabul edilebilir düzeylere geldiği görülecektir.

-Processor: % processor time: Server'da bulunan bütün işlemciler için kullanılıp veri üretebilir. Ayrı ayrı işlemcileri üretip her biri için ayrı ayrı sonuçlar çıkarabileceği gibi sistemin toplam performansı konusunda da değerler sunar. CPU'yu takip etmek için anahtar sayaç bu sayaçtır. Eğer toplam değer %80'i aşarsa sistemde bir işlemci darboğazı olacağından bahsetmeye başlanabilir. İlk 3 aylık dönemin ortalaması %74 olarak gözlemlenmişken, ikinci 3 aylık dönemde bu ortalama %61 olarak ölçülmüştür.

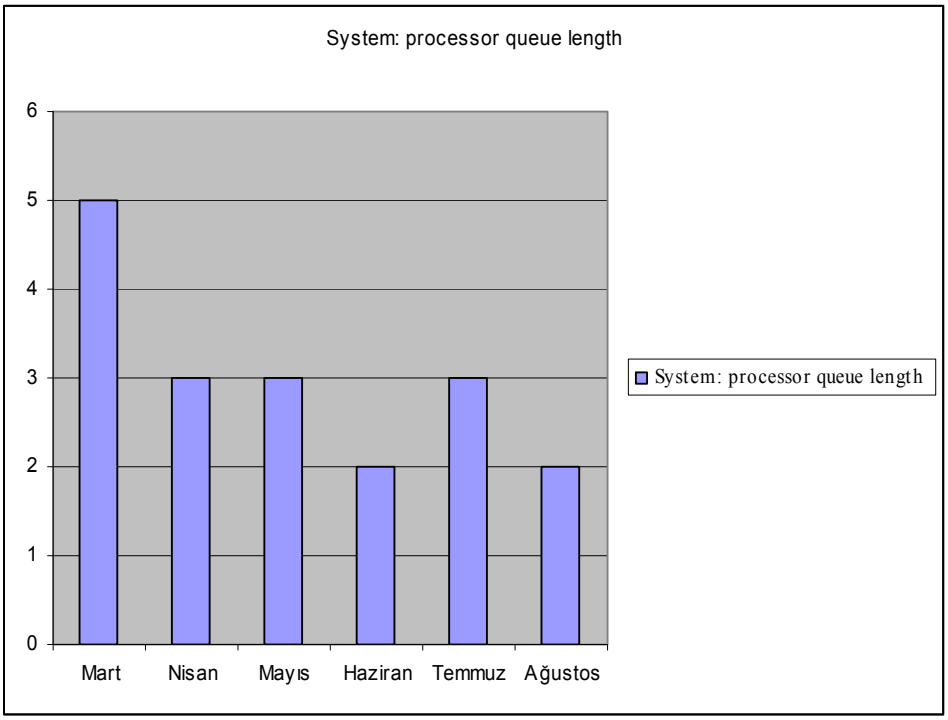
- SQL server buffer: buffer cache hit ratio : Bu sayaç SQL Server'ın veriyi alabilmek için hangi sıklıkla hard diske değil de tampon önbelleğine gittiğini belirtir. OLTP uygulamalarında bu oran %99 ve daha üzeri olmalıdır. Eğer bu oran %90'ın altındaysa derhal RAM alınması zorunludur. Mart ayında bu oran %90'ın bile altındayken ilk 3 aylık dönemin ortalaması 93.6 çıkarak olması gereken %99'luk ortalama değerin altında kalmıştır. Sistemin bellek kaynakları üzerinde darboğazlar oluşmuştur. İkinci 3 aylık dönemde ise bu ortalama %100'de kalmıştır. En düşük seviyesine kampanya dönemi olan temmuz ayında %92 seviyesiyle ulaşmıştır. Ancak ortalama olarak bakıldığı zaman yapılan optimizasyon çalışmaları bu sayaçta da olumlu sonuçlar vermiştir.




Şekil 2.4 Kaynakların kullanımının zamanla değişim grafiği (2)

Şekil 2.4'de değişimi gösterilen sayaç ise memort:available bytes'dır. Bu değer de 5 MB'tan büyük olmalıdır. SQL Server'ın üzerinde koştuğu makinede SQL Server 4–10 MB arasında boş fiziksel bellek arayışındadır. Kalan bellek ise işletim sistemi tarafından

ve SQL Server tarafından harcanır. Eğer kullanılabilir byte miktarın 5MB veya daha az olursa SQL Server'ın bellek yetmezliğinden bir performans sıkıntısı yaşayacağı kesindir. Bu bilgiler doğrultusunda, ortaya çıkan grafik incelenecek olursa, ilk kampanya dönemi olan martta, ortalama 3 MB olarak ölçülmüştür. Bellek ihtiyacı burada kendini göstermiştir. Nisan ve mayıs aylarında ise bu değer sınırda kalmıştır. Tez çalışması boyunca anlatılan bütün optimizasyon maddelerinin uygulanması sonucunda haziran ayında bu değer 8 MB'a yükselerek kabul edilir değerine ulaşmıştır. Sadece temmuz ayında yine kampanya olmasından ve dolayısıyla sunucu aktivitelerinin artmasından dolayı bu değer 6 MB'a düşmüştür. Yine de kabul edilebilir sınır olan 5 MB seviyesinin üzerinde kalmıştır.




Şekil 2.4 Kaynakların kullanımının zamanla değişim grafiği (3)

Bu grafikte ise işlemcide sırada bekleyen thread'lerin ortalama değerleri gözlenmiştir. Eğer işlemci başına sistemde bekleyen işlem sayısı ikiyi geçiyorsa CPU darboğazı ile karşılaşılabilir. Çift işlemciye sahip olan Server'da bu değer 4 sınırında olmalıdır. Ancak görüldüğü üzere yine mart ayında bu değer aşılmış. RAM ihtiyacının yanı sıra CPU darboğazıyla da karşı karşıya kalınmıştır. Nisan ve mayıs aylarında bu değer kabul edilebilir seviyeye gelmiştir. Ancak iyileştirmenin yapıldığı, haziran temmuz ve ağustos dönemlerinin hiç birisinde sınır değer bile gelmemiştir. Yalnızca kampanya dönemi olan temmuzda bu değer, 3'e yükselmiştir ki bu bile kabul edilebilir seviyededir.

Çalışma sonucunda elde edilen iyileştirmeler yalnızca Server'ın kaynaklarının kullanımında olmamıştır. Aynı zamanda t-sql cümleleri ve depolanmış yordamlar üzerinde yapılan değişiklikler sayesinde saha otomasyonu projesinden gelen sorgulara verilen cevap sürelerinde azalmalar gözlenmiştir. Bunun için 6 ay boyunca, sabah 9-11 arasında öğlen 15-17 arasında ve gece 01-03 saatleri arasında Profiler ile veriler toplanmıştır. Profiler'ın izlemesi için Stored Procedures--RPC:Completed , TSQL--SQL:BatchCompleted ana kriterleri verilmiştir. Bu kriterler için, çalıştırılma süresi, yazma işlemi, okuma işlemi, başlama süresi, bitiş süresi verileri gözlenmiştir. Filtre olarak da çalışma süresi 10 saniyeyi geçen sorgular ve depolanmış yordamlar toplanmıştır. Profiler tarafından yakalanan sorguların ilk 3 aylık süre içerinde %12'si bu kritere uymuştur. Mayıs ayı sonrasında depolanmış yordamlar üzerinde ve sorgular üzerinde yapılan, anahtar kelimelerin doğru yerde kullanılması, indekslemenin optimize edilmesi, isimlendirmenin uygun yapılması gibi toplam sorgu süresi optimizasyonu çalışmalarından sonra ( bu çalışmalar 2.6, 2.7, 2.9 bölümlerinde detaylı bir şekilde antalılmıştır.) bu oran % 6'ya inmiştir. Bu çalışmaların yanı sıra bazı spesifik sorguların kontrol edilmesinde ve düzeltilmesinde, araştırma süresinde elde edilen sorgu performansı arttıran kriterler doğrultusunda vb.net koduyla yazmış olduğum "konteskuel" programından faydalanılmıştır.

Tablo 3.10 Önerilen ve ölçülen optimum değerlerin kıyaslanması

| İzlenen Performans Kriteri | En iyi değer olarak beklenen değer | En iyi değer olarak ölçülen değer |
|---|---|---|
| Memeory: Pages/Second | 0-20 arası | Yapılan uygulamalar sonucunda en iyi performans %18'da ölçülmüştür. |
| Physical Disk: % Disk Time | < %55 | Yapılan uygulamalar sonucunda en iyi performans %51'de ölçülmüştür. |
| Proccessor: % Proccessor Time | < %80 | Yapılan uygulamalar sonucunda en iyi performans %64'de ölçülmüştür. |
| SQL server buffer: buffer cache hit ratio | >%99 | Yapılan uygulamalar sonucunda en iyi performans %103'de ölçülmüştür. |
| Memory: available bytes | >5 MB | Yapılan uygulamalar sonucunda en iyi performans 7 MB'da ölçülmüştür. |
| System Proccessor Queue | < 2 (işlemci başına) | Yapılan uygulamalar sonucunda en iyi performans 2 adet işlemde ölçülmüştür. |

Bu çalışma yalnızca SQL Server 2000 için yapılmış ve arada SQL Server 2005 içinde bilgiler verilmiştir. Yapılan diğer çalışmalarla kıyaslandığında [1] [2] [3] [4] [5] [6] jenerik bir veritabanı yönetim sistemlerinde kaynak yönetimi çalışması olmaktan çıkıp, spesifik bir veritabanı yönetim sistemi üzerinde hem kaynak yönetimiyle ilgili hem de sorguların cevap verme sürelerinin kısaltılabilmesi için SQL Server 2000'de performans optimizasyonuna etki edebilecek bütün iç ve dış etkenler incelenmiştir. Benzer konuda çalışma yapılmak istenirse bu çalışmadaki şu eksiklikler giderilmeye çalışılabilir: Çalışmanın tamamı SQL Server 2000 üzerinde yapılmıştır. Ancak araştırma çalışmasına başladığım tarihte SQL Server 2005 henüz piyasada olmadığı için, 2005 üzerinde uygulama ve test yapılmamıştır. MS SQL Server'ın son versiyonu olan 2005 üzerinde bu çalışmalar yapılabilir. Ayrıca maliyet yüksekliğinden ötürü SQL Server donanım performansı denetimi teoride kalmıştır. Burada anlatılanlar pratiğe geçirilerek düzeltmeler, eklemeler yapılabilir. Son olarak vb.net koduyla yazılmış olan konteskuel programının kodu geliştirilebilir. Program girdi olarak verilen bazı sorgulara alternatif sorgu önerisinde bulunamamaktadır. Mümkün olduğunca çok durum göz önüne alınarak verilen sorguların büyük çoğunluğuna alternatif üretmesi sağlanabilir.

# BÖLÜM 1. GİRİŞ

MS SQL Server 2000'ın kullanımda istenilen performansı sağlayabilmesi için düşünülmesi gereken birçok performans ölçütü bulunmaktadır. SQL Serverın koştuğu makinenin donanımsal ayarları, üzerinde çalıştığı makinede bulunan işletim sisteminin ayarları, SQL Serverın ve üzerindeki veritabanlarının ayarları, SQL Server üzerindeki veri tabanlarında kullanılan indekslerin ayarları, kullanılan t-sql cümleciklerinin doğru yazılması, SQL Server üzerinde çalışan görevlerin (Job) doğru zamanlanması, doğru oluşturulması, üzerinde düşünülmesi gereken çok ayrıntılı ana maddelerdir. Bu maddeler üzerinde kontrol listeleri oluşturulup, bu maddelerin canlı ortamlar üzerinde uygulandıktan sonra bu kontrol listelerinin doldurulması bile SQL Serverın performans yönetimi açısından yeterli olmayacaktır. Bütün bu yapılanların server üzerindeki etkileri performans görüntüleme araçları ile takip edilip, Profiler aracıyla SQL Server'ın yapılan değişikliklere verdiği tepkiler canlı ortamlar üzerinde izlenilmeli, elde edilen veriler değerlendirilmeli ve yapılan performans arttırma çalışmaları, elde edilen kontrol listelerine göre gerekiyorsa yeniden düzenlenmelidir.

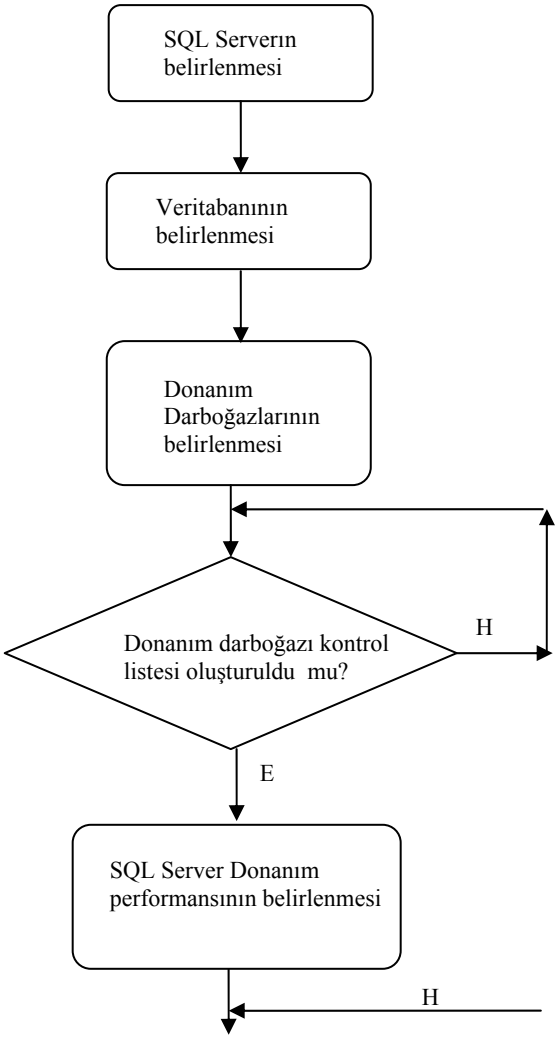
Veritabanları üzerindeki performans sıkıntıları veritabanları büyüdükçe uğraşması gittikçe zorlaşan bir konu haline gelir. Geçmiş yıllarda sorunların teşhis edilmesi ve bu sorunların giderilmesi konusunda çalışmalar yapılmıştır [1-6]. Bu çalışmaların üzerinde durdukları konu veritabanları performans yönetimi sırasında sistem kaynaklarının kullanımı üzerine olmuştur. Bu çalışmalara göre; veritabanı yönetim sistemleri ayarlamaları iki ana konuyla ilgilenir: teşhis ve kaynak ayarlaması. Teşhis, hangi kaynağın soruna sebep olduğu ile ilgilenirken, kaynak ayarlaması daha iyi performans için kaynakların ilgili işlemler arasında paylaştırılması konusuyla ilgilenir [6] .

Bu çalışmanın yapılmasının maksadı da MS SQL Server 2000 sunucusu üzerinde performansa yönelik olan bütün iç ve dış ayarların incelenmesidir. Diğer çalışmalar genel veritabanı yönetim sistemleri üzerinde dururken bu çalışma spesifik olarak SQL Server 2000 üzerinde durmuştur. Çalışma, Server kaynaklarının kullanımının optimize edilmesi ve sorguların optimize edilmesi konseptleri içerisinde incelenmiştir. Performansı etkileyen bütün seçenekler tek tek uygulandığında server performansı üzerinde yaptığı etkilerin incelenmiş, uygulanmış, performans ölçütleriyle ilgili jenerik kontrol listelerinin oluşturulmuş ve optimum çözümlerin bulunması sağlanmıştır. Çapraz kullanımları sonucunda veritabanı sunucusunun çökmesine kadar ciddi sonuçları olacak bu ayarların her birisinin birbiri üzerindeki etkileri ve server üzerindeki etkileri tek tek incelenmiştir.

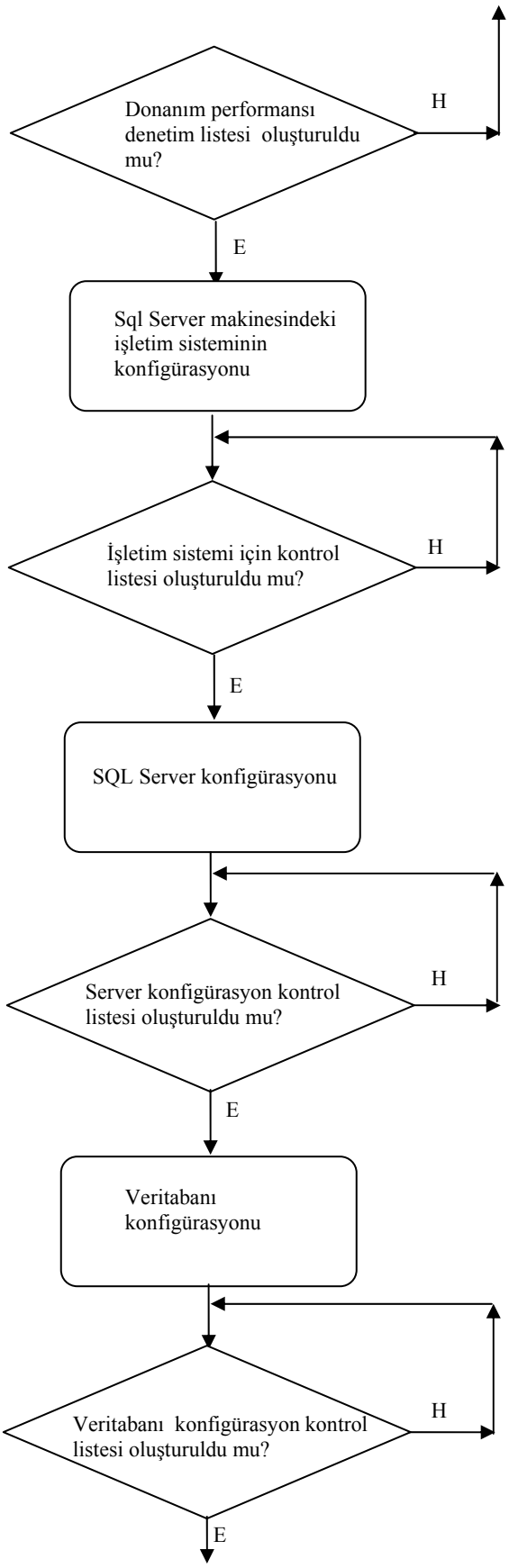
Bu çalışma yapılırken SQL Server 2000 performansını etkileyen bütün iç ve dış faktörler listelenmiş, her birinin ayrıntılı etkileri ortaya çıkarılmıştır. Test süreci 6 ay 24 saat boyunca 400 kullanıcı tarafından kullanılan bir saha otomasyonu projesi üzerinde gerçekleşmiştir. Yapılan değişikliklerin server üzerindeki etkileri incelenmiştir ve çalışmanın sonuç bölümünde her bir ayarın bu veritabanı sunucusu üzerindeki etkileri verilmiştir.

SQL Server performans ölçütlerinden birisi olan sorgu optimizasyonu konusu, veritabanı üzerindeki tablolarda kullanılan indekslerden, yazılan t-sql cümlesinde kullanılan sentakstan etkilenen bir kriterdir. Sentaks üzerinde veya indeksler üzerinde yapılan değişikliklerden bazıları hemen yeni çalışma planlarının oluşturulmasına ve sorgu sonuçlarının sürelerinin uzamasına ya da kısalmasına sebep vermektedir. Bu sorgu sonucu süresine etki eden bütün kriterlerin bir yazılımcı yada veritabanı yöneticisi tarafından düşünülmesi mümkün olsa da oldukça zor ve ayrıntılı bir iştir [1]. İşte bu çalışmanın ana amaçlarından bir tanesi de bu performans kriteri için uygun bir sorgu performansı kontrol programı oluşturulmasıdır. Çalışmanın 2.6, 2.7, 2.9 ve 3. bölümlerinde incelenen konular doğrultusunda, bir sorgu performansı kontrol programı

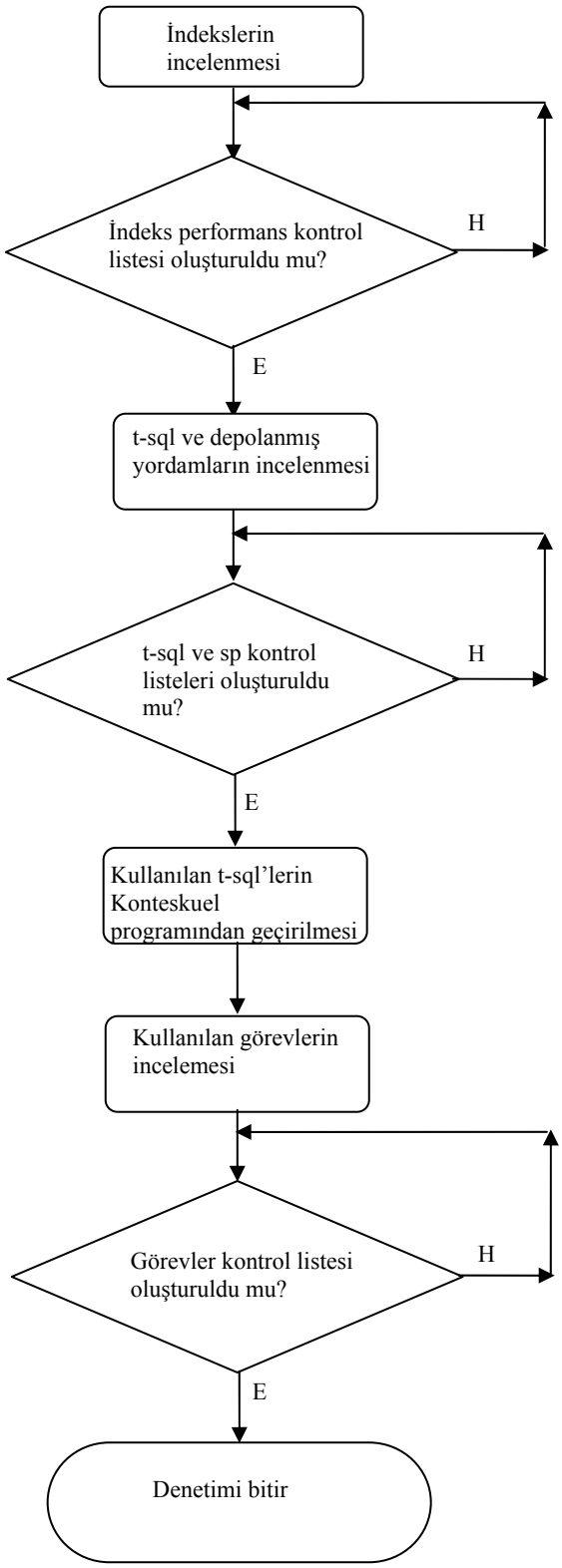
yazılmıştır. Bu programın amacı, yazılan t-sql cümleciklerini kontrol ederek, bu bölümlerde bahsedilen ölçütlere göre çapraz değerlendirmelerini yapıp, yanlış kullanılan ayrılmış kelimelerin (reserved words ), birleştirme işlemlerinin (join operators) kullanılmasını önlemektir. Program yazılan bir t-sql cümleciğinin yapısını inceledikten sonra, bu sorgunun yerine kullanılacak olan performans açısından daha olumlu bir sorgu var ise bunu önermektedir. Tez çalışması boyunca aşamasında izlenen model yol aşağıdaki akış diyagramında belirtilmiştir;




Şekil 1.1 Denetim çalışması akış diyagramı




Şekil 1.1 Devam




Şekil 1.1 Devam

# BÖLÜM 2. SQL SERVER PERFORMANS DENETİMİNİN GERÇEKLEŞTİRİLMESİNDE İZLENEN YOLLAR

Veri tabanları üzerinde uzunca süreler çalışmış ve değerlendirme de bulunan kişiler veri tabanı performans yönetimi işinin bir bilim gibi işlemediğini bir doğrunun her koşulda doğru olmadığını değişik durumlara göre alınacak önlem ve uygulanacak işlemlerin farklı olabileceğini bilirler. Her zaman en optimum çözümü bulmak birçok kritere bağlıdır. Mesela bir tane performans arttırma yöntemi performansı bir yerden arttırmaya sebep olurken aynı zamanda başka bir yerde performansı olumsuz yönde etkileyip toplamda performansı aşağıya çekebilir [2].

Bu denetim serisinin amacı SQL Server üzerindeki performans problemlerini belirlemek ve bunları ortadan kaldırıcı ve performansı yükseltici yaklaşımlar ortaya koymaktır. Bu denetim listelerini hazırlamanın ve uygulamanın faydası sadece performansı yükseltmek için ne yapılabileceğinin görülmesi değil aynı zamanda hali hazırda sistemin durumunun ne olduğunu nelerin yanlış nelerin doğru konfigüre edildiğini görülmesini de sağlamasıdır. Bazı durumlarda uygulanması gerekenler alınması gereken önlemler bu denetim dizisinde anlatılacaklardan tamamen farklı da olabilir. Çünkü SQL Server performansı artırımı durumdan duruma göre değişiklik gösterebilecek bir durumdur. Buradaki dizide ve bu tez boyunca anlatılacak özellikler ise en geniş kapsamdaki durumları kapsamayı hedeflemektedir.

İdeal olarak bu performans denetiminin çalışılan bütün serverlar üzerinde periyodik olarak yapılmasıdır. Zaman problemiyle karşılaşıldığı zamanlarda da önceliğin en fazla

performans problemin yaşandığı ve görev kritikliğine haiz olan serverlara verilmesi önerilir.

SQL Server performans denetimini en doğru ve yönetilebilir şekilde yapabilmek için bu bölümde izlenilecek olan adımları şu şekilde sıralamak doğru bir yaklaşım olacaktır:

- SQL Server donanım darboğazlarını belirlemek için performans görüntüleme aracının kullanılması.

- SQL Server donanım performansı kontrol listesi oluşturulması.

- İşletim sistemi performansı kontrol listesi oluşturulması.

- SQL Server 2000 konfigürasyonu performansı kontrol listesi.

- Veritabanı konfigürasyon ayarları performansı kontrol listesi.

- İndeks performansı kontrol listesi.

- Uygulama ve t-sql performansı kontrol listesi.

- SQL Server veritabanı iş (JOB) performansı kontrol listesi.

- Düşük performans gösteren sorguların belirlenmesi için Profiler kullanılması.

- En iyi SQL Server performans denetiminin gerçekleştirilmesi.

SQL Server performans denetimi için yukarıdaki maddeleri bir bir ele alıp her biri için gerekli işlemleri yapıp denetim listeleri oluşturup bu denetim listelerini bir baz alma noktası olarak kullanıp o zamandan sonraki denetimler için kullanmak iyi olacaktır.

## 2.1. Veritabanı Performansı Takip Edilirken Kullanılan Yol Haritası

Bölüm 3 boyunca anlatılacak olan konuların teknik olarak çok detaylı olmasından dolayı araştırmanın anlaşılabilirliğini kolaylaştırmak açısından yapılması gereken bazı tanımlamalar ve açıklamalar vardır. Bölüm 3'de anlatılacak olan ana başlıkları tek tek inceleyip bu çalışmaların yapılmasında neyin amaçlandığı ve bu başlıklarda kullanılan yöntemlerin temel bilgileri bu bölümde verilecektir.

Bu araştırma çalışması boyunca bütün bu performans ölçütleri detaylı bir şekilde incelenmiştir. Her bir ölçütün beklenen değerleri ortaya konulmuştur. Bunun da ötesine geçip bu performans kriterleri 6 ay boyunca saha otomasyonu projesinin veritabanında test edilerek , önerilen eşik değerleriyle benim bulduğum eşik değerleri karşılaştırılmıştır. Bu karşılaştırmalar grafikleriyle birlikte sonuç bölümünde sunulmuştur.

### 2.1.1. SQL Server veritabanı donanım performansı kontrolü temelleri

Çalışmaların maliyeti açısından bakıldığı zaman, en yükse gider kalemi bu performans çalışmasında görünmektedir. Bu ana başlık altında amaçlanan ana konu, SQL Server'ın üzerinde koştuğu makinenin donanım kriterlerinin hangi özelliklerde olması gerekliliğidir. Buna etki eden temel bileşenler CPU, RAM ve sabit disktir. Bu ana bileşenlerin hangi niteliklere sahip olması gerekliliğinden, eşik değerlerinden, bu eşik değerleri aşıldığı zaman yapılması gereken işlemlerden bu bölümde bahsedilecektir.

### 2.1.2. SQL Server işletim sistemi performans denetimi temelleri

Unutulmamalıdır ki SQL Server bir işletim sistemi sunucusu değil, veritabanı sunucusudur. Doğal olarak da üzerinde koştuğu makinedeki işletim sisteminin özelliklerinden etkilenmektedir. İşletim sisteminin seçimi, işletim sisteminin kullandığı dosyalama sistemlerinin (NTFS) seçimi, yayınlanan güvenlik paketlerinin kullanılıp kullanılmadığının kontrolü, SQL Server'a etki edecek olan servislerin çalışıp çalışmadığının, çalışanların birbiriyle çakışıp çakışmadığının belirlenmesi maddeleri, SQL Server'ın performansını dolaylı yönden etkilemektedir. Bu maddeler aslında SQL Server'ın koştuğu makinenin toplam performansını doğrudan ilgilendirir. Ancak SQL Server'da bu paylaşımdan bir fayda görür.

### 2.1.3. SQL Server konfigürasyonu performans denetimi temelleri

MS SQL Server veritabanında, sunucu ve veritabanı konfigürasyonu olmak üzere birbirine çok benzeyen iki farklı kavram vardır. Bu kavramları birbirinden ayırt etmek önemlidir. SQL Server konfigürasyonu, o sunucu altında bulunan bütün veritabanlarına etki eder. Veritabanı konfigürasyonu ise sadece o veritabanı için yapılan konfigürasyon ayarlarıdır.

Bu konu başlığında, server üzerinde çalışan veritabanlarının hangi durumlarda paralelliği kullanacakları, imleç kullanılması halinde performans ayarının nasıl yapılması gerektiği, doluluk oranlarının neye göre ayarlanması gerektiği, sorgu başına kullanılacak minimum bellek miktarları gibi uygulanması halinde, sunucunun üzerinde çalışan bütün veritabanlarına etki edecek olan ayarlar araştırılmıştır. Beklenen değerler ortaya konulmuştur. Hangi değerin neyi ifade ettiği, o değerlerin altında ya da üstünde sonuç çıkması halinde alınacak önlemler açıklanmıştır.

### 2.1.4. SQL Server veritabanı ayarları performans denetimi temelleri

Sunucu üzerinde çalışan veritabanlarının bireysel ayarlarının yapılması için anlatılan bir konudur. Buna göre veritabanının performansına etki eden, istatistiklerin otomatik olarak oluşturulması, son kullanıcı sistemden çıktığı zaman sistemin otomatik olarak kapatılması, yırtık sayfa belirlenmesinin ve onarımının otomatik olarak yapılması, otomatik veritabanı büyüme ve küçülme oranlarının belirlenmesi gibi veritabanı ayarlarını bu ana başlık altında tüm detaylarıyla ele alınmıştır. Bu değerlerde hangi kriterleri hangi durumlarda kullanmak gerektiğinden bahsedilmiştir.

### 2.1.5. SQL Server veritabanı indeks performansı denetimi temelleri

Dünyada veritabanı sistemlerinin kullanım amaçları iki ana başlık altında toplanmaktır. Verinin depolanması ve işlenmesi. Gerek verinin depolanmasının gerekse de işlenip çıktı alınmasının hızlı olması, verinin doğruluğu kadar önemlidir. İndeks verilerin depolanmasında ve raporlanmasında çok önemli bir etkiye sahiptir.

Çok veri girişi olan ama buna karşılık çok fazla arama olamayan bir tabloda kullanılacak olan indeksleri, veri girişiyle, veri arayışının aynı derecede önemli olduğu bir tabloda kullanılan indeks türleri, veri girişinin nadir olduğu ancak büyük verilerin girişinin olduğu, çok sıklıkla veri aramasının yapıldığı tablolarda kullanılması gereken indeksler bu konu başlığı altında incelenmiştir.

Aynı zamanda bir tabloda bulunan indekslerin birbirleriyle uyumlu olarak bir arada bulunup bulunmadığının belirlenmesi, fazla veya eksik indekslerin belirlenmesi, tablolarda kullanılmayan indekslerin olup olmaması gibi ayrıntılar da araştırmanın bu konu başlığı altında incelenmiştir.

### 2.1.6. T-SQL kontrol listelerinin temelleri

Veritabanı sistemi hangi üretici firmadan olursa olsun hepsinin kendine göre özelleştirdiği bir veritabanı sorgu sistemi vardır. ANSI sql bütün veritabanı sistemlerinde doğru bir şekilde çalışması beklenen bir sistemdir. Ancak Microsoft bunu kendine göre özelleştirmiş, zenginleştirmiş ve bunu "t-sql" haline getirmişti.

T-sql cümlecikleri bir veritabanı sisteminden veri çekişi veya eklenmesi veya güncellenmesi için kullanılan cümleciklerdir. Görüldüğü üzere veritabanı üzerinde yapılan en hayati işlemler aslında programatik olarak t-sql'ler yardımıyla yapılmaktadır. Onun için bu sentaksın en doğru şekilde kullanılması, aynı sonucu verecek olmasına rağmen sistemi gereksiz yere yoracak cümlelerden kaçınılması son derece önemli ve karışık bir operasyondur. Bu ana başlık altında nelere dikkat edilmesi nelerden kaçınılması konusunda bilgiler verilmiştir.

Araştırma boyunca bahsedilen indeks performansı denetim listeleri ve t-sql kontrol listeleri yardımıyla "konteskuel" adında vb.net ile bir sorgu iyileştiricisi programı yazılmıştır.

### 2.1.7. İzleyici (Profiler) kullanımı

Araştırma boyunca araştırılan, performans eşik değerleri ortaya koyulan ölçütlerin veritabanı sistemi üzerinde gerçek uygulama sistemi üzerinde nasıl etki edeceği pro-aktif olarak belirlenmelidir. Bu da canlı ortamın aynısının bir test ortamına taşınması, beklenen yükün sunucuya uygulanması ve de oluşacak sonuçların gözlenmesiyle mümkündür. Aksi taktirde bütün uygulamalar hiçbir testten geçmeden canlı ortamlarda uygulanacak olursa beklenmedik sonuçlar doğurabilir.

Yapılan değişikliklerin server üzerinde yaptığı değişiklikler, işlemciye bindirdiği yük, belleğe bindirdiği yük, sabit disklerin ve disk dizilerinin çalışmasının, yapılan değişikliklere verdiği cevaplar, değiştirilen t-sql cümleciklerinin çalışma ve cevap verme sürelerinin karşılaştırılması gibi ölçümler yapılan değişikliklerin somut olarak gözlenmesi olanağını sunar. MS SQL Server 2000 üzerinde varsayılanda gömülü olarak gelen izleyici (Profiler) bütün bu işlemlerin yapılmasını, eşik değerlerinin, filtrelerin, izlenecek performans ölçütlerinin özelleştirilmesini sağlayan ve verileri toplayan bir araçtır. Bu bölüm boyunca bu aracın nasıl kullanılması gerektiği ve sonuçların nasıl değerlendirilmesi gerektiği ortaya koyulmuştur.

### 2.1.8. Konteskuel programının kullanımı

Araştırmamın bir parçası olarak yazdığım ve açık kullanıma sunduğum konteskuel programının ne amaçlanarak yazıldığı, çalışma mekanizmasının ne olduğu, nasıl kullanılması gerektiği, çıkan sonuçların nasıl değerlendirilmesi gerektiği bu bölümde gerek metinsel gerekse görsel olarak sunulmuştur.

# BÖLÜM 3. SQL SERVER PERFORMANS DENETİMİNİN GERÇEKLEŞTİRİLMESİ

Bölüm 2'de genel hatlarıyla bahsedilen performans denetimi çalışmaları bu bölüm boyunca detaylı bir şekilde anlatılacaktır. Bu bölümde her bir performans kısıtı için incelenmesi gereken özellikler, bu özelliklerin [8-11] değerleri, karşılaşılan değerlerin sistemle ilgili verdiği mesaj, karşılaşılan darboğazlardan kurtulma yöntemleri verilmiş, yapılan araştırmalar sonucunda performansa en çok etki yapan maddeler listelenmiş ve denetim aşamasında bunların kullanılması için denetim listeleri oluşturulmuştur.

## 3.1. SQL Server Donanım Darboğazlarını Belirlemek İçin Performans Görüntüleme Aracının Kullanılması

Tablo 3.1 Donanın darboğazları denetimi kontrol listesi

| Sayaç Adı | Ortalama | Minimum | Maksimum |
|---|---|---|---|
| Memory: Pages/sec (Bellek:Sayfa/saniye) | | | |
| Memory: Available Bytes (Bellek:Kullanılabilir Byte Miktarı) | | | |
| Physical Disk: % Disk time (Fiziksel disk: (Disk Zamanı Yüzdesi) | | | |
| Physical Disk: Avg. Disk Queue Length (Fiziksel disk: Ortalama Disk Sırası Uzunluğu) | | | |
| Processor: % Processor Time (İşlemci:İşlemci zaman yüzdesi) | | | |

Tablo 3.2 Donanın darboğazları denetimi kontrol listesi (Devam)

| | | | |
|---|---|---|---|
| System: Processor Queue Length (Sistem: İşlemci sıra uzunluğu) | | | |
| SQL Server Buffer: Buffer Cache Hit Ratio (SQL Server Tamponu: Tampon Ön Bellek Erişim Miktarı) | | | |
| SQL Server General: User Connections (SQL Server Genel: Kullanıcı Bağlantıları) | | | |

SQL Server performans denetimine başlamak için en uygun yer performans görüntüleme (sistem görüntüleme) aracı olabilir. 24 saat boyunca seçilen sayaçlara göre oluşturulmuş bir rapor SQL Serverın karşılaştığı donanımsal sıkıntıları belirlemek için çok iyi bir başlangıç olacaktır.

24 saatlik veriyi performans görüntüleme aracından aldıktan sonra Tablo 3.1'de verilen sayaçlar seçilerek grafikleri görüntülenebilir. Değerler normal değerleriyle karşılaştırıldığı zaman potansiyel tehlikeler darboğazlar kolaylıkla belirlenebilir.

### 3.1.1. Seçilen performans sayaçlarının sonuçların yorumlanması

Aşağıda önemli performans görüntüleme aracı sayaçlarının tavsiye edilen değerlerinden ve donanımsal darboğazları çözmek için yapılması önerilen adımlardan bahsedilmektedir.

#### 3.1.1.1. Bellek:sayfa/saniye

Bu sayaç , RAM'den diske saniyede geçen sayfa sayısını belirtir. Ne kadar fazla sayfa geçişi olursa server üzerinde o kadar fazla I/O yükü biner. Bu da performansı olumsuz yönde etkiler. Amaç bu geçişi elimine etmek değil minimize etmek olmalıdır.

Eğer SQL Serverın koştuğu makine üzerinde çalışan tek ciddi uygulama SQL Server ise bu sayacın yüzdesi 0 ile 20 arasında olmalıdır. Buradaki anahtar cümle ortalama sayfanın %20'den az olması gerekliliğidir. Eğer bu oran %20den fazlaysa bunun sebebi RAM gerekliliği olabilir. Genel söylenti şöyledir; sistem dene kadar çok RAM var ise sayfalama işlemi o kadar az olur.

Birçok durumda SQL Serverın koştuğu ve yeterli miktarda RAM'e sahip olan bir makinenin sayfalama yüzdesi %20'nin altındadır. Yeterli miktarda RAM'den anlaşılması gereken de; tampon önbelleği kullanım seviyesi %99 veya daha fazla olan belleklerdir. Eğer Buffer Hit Cache ratio'su yani tampon önbelleği kullanım seviyesi %99 veya daha üzeri bir RAM çalışmakta ve buna rağmen bu sayaç %20 civarlarında geziniyorsa o makine üzerinde koşan ve RAM ihtiyacı çok olan başka bir uygulama daha var mı diye bakmak gereklidir. Eğer bu tarz bir durum var ise SQL Server'ın maksimum performansta çalışması için o makinede çalışan tek ana program olması sağlanmalıdır.

Başka bir program olmamasına rağmen %20 civarlarındaysa ve hatta bu oran aşılmışsa SQL Server bellek ayarları gözden geçirilmelidir. SQL Server tekrar konfigüre edilerek "SQL Server belleğini dinamik olarak ayarla" seçeneği seçilmelidir. En optimum performans için "Maksimum Bellek" ayarı en yüksek seviyeye çıkarılmalıdır. En

optimum performans için SQL Server ihtiyaç duyduğu zaman RAM'den istediği kadar kaynak alabilmelidir.

#### 3.1.1.2. Bellek : Kullanılabilir byte miktarı

SQL Serverın yeterli RAM'i olup olmadığını kontrol etmenin bir diğer yolu da Bellek:Kullanılabilir Byte'lar sayacını gözetim altına almaktır. Bu değer de 5 MB'dan büyük olmalıdır. SQL Server'ın üzerinde koştuğu makinede SQL Server 4–10 MB arasında boş fiziksel bellek arayışındadır. Kalan bellek ise işletim sistemi tarafından ve SQL Server tarafından harcanır. Eğer kullanılabilir byte miktarın 5MB veya daha az olursa SQL Server'ın bellek yetmezliğinden bir performans sıkıntısı yaşayacağı kesindir. Bundan kurtulmak için ya Server'ın RAM'ini yükseltmek gerekir , ya bir şekilde server üzerindeki yükü azaltmak gerekir ya da yapılabiliyorsa SQL Serverın bellek konfigürasyon seçeneklerini değiştirmek gerekir.

#### 3.1.1.3. Fiziksel disk: % disk zamanı

Bu sayacın ölçtüğü değer ise fiziksel bir dizinin (bu belirli bir disk dizisi veya diskin mantıksal bölümü değildir) ne kadar meşgul olup olmadığıdır. Dizilerin ne kadar meşgul olduğunu görmek için güzel bir karşılaştırma verisi verir.

Bir kural olarak % Disk Time sayacı %55'den daha küçük olmalıdır. Eğer sürekli olarak bu yüzdenin üzerinde bir seviyede bulunuyorsa (10 dakika ve bu süreyi aşan zamanlarda sıkıntı vardır denebilir) SQL Server'da bir darboğaz oluşacak denilebilir. Eğer bu 24 saatlik bir zaman diliminde yalnızca 1–2 defa olan bir durumsa endişelenecek çok büyük bir durum yoktur ancak bu sayaç sürekli olarak bu seviyelerde geziniyorsa o zaman Server'ın I/O performansını yükseltecek bir çözüm aranmaya başlanmalıdır. Bunlardan

bazıları yeni bellek eklemek, daha hızlı sürücülerle değiştirmek, kontrolcü kartına ek önbellek tahsis edilmesi, farklı bir RAID kullanılması olabilir.

#### 3.1.1.4. Fiziksel disk: ortalama disk kuyruğu uzunluğu

% Disk süresi sayacını izlemenin yanı sıra ortalama disk kuyruğu uzunluğu sayacı da takip edilmelidir. Devam eden süreçlerde, disk dizindeki bütün diskler için 2 değeri aşılıyorsa (10 dakikayı aşan sürelerde) ortada bir I/O darboğazı potansiyeli vardır. Physical disk: % disk time sayacı gibi bu olay 24 saatlik bir süreçte sadece 1–2 defa olduysa endişe edilecek çok bir şey yoktur. Ancak çok sıklıkla oluyorsa I/O performansını artırıcı çözümler aranmaya başlanmalıdır.

Bu durumu manuel olarak hesaplamak gerekecektir. Örneğin, 6 fiziksel dikten oluşan bir dizi var ise ve ortalama disk sırası uzunluğu her bir disk için 10 ise bu durumda gerçek disk sırası uzunluğu 10/6 =1.66'dır ki bu değer de 2 eşik seviyesini geçmiyordur.

Server'ın herhangi bir I/O darboğazıyla karşılaşıp karşılaşmayacağını anlamak için hem % disk time sayacı hem de avg. disk queue length sayacı birlikte değerlendirilmelidirler. Örneğin % disk time sayacının % 55'in üzerinde olduğu zamanlar çok görülüyor ise ve aynı zamanda avg. disk queue length sayacı disk başına 2 değerini aşmışsa bir I/O darboğazı yaşanacağı kesindir.

#### 3.1.1.5. İşlemci: % işlemci süresi

İşlemci Nesnesi: % İşlemci Süresi sayacı Server'da bulunan bütün işlemciler için kullanılıp veri üretebilir. Ayrı ayrı işlemcileri üretip her biri için ayrı ayrı sonuçlar

çıkarabileceği gibi sistemin toplam performansı konusunda da değerler sunar. CPU'yu takip etmek için anahtar sayaç bu sayaçtır. Eğer toplam değer %80'i aşarsa sistemde bir işlemci darboğazı olacağından bahsetmeye başlanabilir. Eğer bu tarz durumlar arada sırada oluyor ve meydana geliş sebepleri biliniyorsa sorun oluşmayabilir. Ancak sıklıkla meydana geliyorsa ya daha hızlı işlemci alarak, ya işlemci sayısını arttırarak ya da daha geniş on-board L2 önbelleğine sahip işlemciler alınarak bu sorunun üstesinden gelinebilir.

#### 3.1.1.6. Sistem: işlemci kuyruğu uzunluğu

İşlemci zamanı yüzdesi sayacıyla birlikte işlemci sırası uzunluğu sayacının sonuçlarını da bilmek gerekir. Eğer işlemci başına sistemde bekleyen işlem sayısı ikiyi geçiyorsa CPU darboğazı ile karşılaşılabilir. Örneğin sistemde 4 tane işlemci çalışmakta. O zaman bu değer 8'i geçmemelidir.

Eğer sırada bekleyen işlem sayısı sürekli olarak eşik değeri olan 2 ortalamasını geçiyor ancak CPU çalışma yüzdesi çok yüksek yüzdelerde görünmüyorsa SQL Server için "max. worker threads" konfigürasyon ayarının değiştirilmesi gerekmektedir. Böyle bir durumda işlemci sırası uzunluğunun yüksek olmasının sebebi ise sırada bekleyen "worker thread" olarak bilinen işlemlerin sayısının yüksek oluşudur. "maximum worker threads" ayarında sayı düşürülerek thread havuzlaması yapılır ve bu şekilde değerler tekrar istenilen seviyelere yükseltilebilir.

Sistemde bir işlemci darboğazı olup olmadığını daha sağlıklı bir şekilde belirlemek için işlemci kuyruğu uzunluğu ve % toplam işlem süresi sayaçları bir arada kullanılmalıdır. Eğer her iki sayaç da eşik değerlerini aşıyorsa bir CPU darboğazıyla karşılaşılacağı beklenmelidir.

### 3.1.1.7. SQL server ara belleği : ara bellek kullanım miktarı

Bu sayaç SQL Server'ın veriyi alabilmek için hangi sıklıkla hard diske değil de tampon önbelleğine gittiğini belirtir. OLTP uygulamalarında bu oran %90'ı aşmalıdır ancak ideal olan değer %99 ve daha üzeridir. Eğer bu oran %90'ın altındaysa derhal RAM alınması zorunludur. Kaldı ki bu yüzdenin %90 ile %99 arasındaki değerlerinde bile RAM takviyesine ihtiyaç vardır. Bazı durumlarda eğer veritabanınız çok büyükse %99danbiraz daha düşük yüzdeler kabul edilebilir.

OLAP uygulamalarında ise bu oran daha da düşük olabilir. Ama gerek OLAP gereksek OLTP uygulamaları için koşan SQL Serverlar bu sorunla karşılaşacak olursa RAM takviyesi sorunu çözecektir.

### 3.1.1.8. SQL server genel: kullanıcı bağlantıları

Bu sayaç SQL Server'a yapılmış olan bağlantı sayısını verir. Eğer bu sayı 255'i geçerse SQL Server konfigürasyon ayarlarındaki "Maximum Worker Threads" ayarını varsayılan değeri olan 255'den daha yüksek bir değere yükseltilebilir. Her zaman için "Maximum Worker Threads" ayarındaki sayı sisteme bağlı olan bağlantı sayısından yüksek olmalıdır. Aksi halde performans konusunda sıkıntı yaşanır.

## 3.2. SQL Server Donanım Performansı Kontrol Listesi

Tablo 3. 3. Performans denetimi kontrol listesi

| SQL Server Donanım Özellikleri | Değerler |
|---|---|
| İşlemci Sayısı | |
| CPU MHz | |
| CPU L2 Önbellek Miktarı | |
| Fiziksel RAM Miktarı | |
| Serverdaki kullanılabilir boş disk miktarı | |
| Her bir dizideki fiziksel disk sayısı | |
| SQL Server için kullanılan RAID seviyesi | |
| Donanım yazılım RAID karşılaştırması | |
| Disk Bölünme Seviyesi | |
| İşletim sisteminin yeri | |
| SQL Server çalıştırılabilir dosyalarının yeri | |
| SQL Server takas dosyalarının yeri | |
| Tempdb veritabanının yeri | |
| Sistem veritabanlarının yeri | |
| Kullanıcı veritabanlarının yeri | |
| Kayıt dosyalarının yeri | |
| Serverdaki disk kontrolcüsü sayısı | |
| Serverdaki disk kontrolcülerinin tipi | |
| Serverdaki disk kontrolcülerinin önbellek miktarları | |
| Disk kontrolcüsündeki Önbelleğe Geri Yaz (Write Back Cache) ayarı açık mı kapalı mı? | |
| Disk sürücülerinin hızı | |
| Serverda kaç tane ağ kartı vardır? | |

SQL Serverın donanım özelliklerinin denetlenmesi ilk başlarda yapılmalıdır. Bu bölümde denetim yapılması anlatılacak olan donanımları şu ana başlıklar altında toplanabilir.

-İşlemci

-Bellek

-Disk

-Ağ bağlantısı

-Diğerleri

Aşağıda anlatılacak olan denetimler sonucunda yukarıdaki tablodaki soruların cevapları verilerek potansiyel tehlikeler öngörülebilir.

### 3.2.1. İşlemci

Bu bölüm aslında oldukça açıktır. Sistemin desteklediği sınırlar ölçüsünde, sistemde ne kadar çok ve yüksek işlem hızına sahip işlemci olursa SQL Serverın çalışması da o kadar hızlı olacaktır.

Veri tabanı olarak SQL Server ile çalışan herhangi bir uygulamanın kaç tane işlemciye ihtiyaç duyacağını söylemek zordur. Bunun sebebi; her bir uygulama farklı bir performansa ihtiyaç duyması ve SQL Server'ı kullanma oranının bir diğerine göre farklı olabilmesidir.

Kaç adet işlemci alınacağının belirlenmesindeki zorluk sebebiyle aşağıdaki şu maddeler bir öngörü yapılabilmesi için yardımcı olabilecek niteliktedir:

-Alınacak bir Server'ı mümkün olduğunca çok işlemcili almak kesin çözüm olacaktır.

-Eğer yukarıdaki madde gerçekleştirilemiyorsa alınacak Server'da ek işlemciler takılıp yükseltmeye olanak sağlayacak yuvaların bulunması ilerisi için güzel bir planlama olacaktır. Hemen her SQL Server zaman geçtikçe daha fazla işlemciye ihtiyaç duyar.

Bazı muhtemel senaryoları ve olması gereken yaklaşımları şu maddeler altında toplamak mümkündür;

-Özel bir muhasebe programının kullanımı için bir SQL Server ayarlanabilir. Bu program ilerideki bir kaç yıl için yalnızca 5 kullanıcıya açık olacaktır. Bu tarz bir durumda uygulamanın yaptığı işlemlere göre değişmekle birlikte genel öngörü 1 CPU'nun bu durum için yeterli olacağıdır. Beklentilerin tersine kullanıcı sayısı artar ya da uygulama SQL Server üzerinde çok fazla işlem yaptığı için beklenenden daha çok kaynağa ihtiyaç duyulduğu için işlemciye ihtiyaç duyulabilir. Bu tarz bir durum için mutlaka ana kart üzerinde ikinci bir CPU için genişleme yuvası gerekecektir.

-Sadece OLTP işlemlerinin olacağı ve raporların üretileceği bir uygulama yazıldığı varsayılsın. Bu uygulama rapor üreteceği zaman SQL Server'ı bir miktar yoracak. Uygulamaya aynı anda maksimum 25 kullanıcının bağlandığı varsayılsın. Bu tarz bir senaryoda 2 CPU'ya ihtiyaç duyulabilir. Ancak daha fazla işlemciye ihtiyaç duyulup duyulmayacağı kestirilemez. Çünkü " bir miktar yoracak raporlar" cümlesi başa dert açabilecek nitelikte bir cümledir. Bu durumda 2 işlemci tabanlı olarak tasarlanmış SQL Server'ın en az 2 tane daha genişleme yuvasına sahip olması, olası negatif senaryolar için iyi bir bekleme stratejisi olacaktır.

-SQL Serverın aynı anda 100–150 kullanıcının kullanabileceği bir ERP programı için koştuğu düşünülecek olursa en az bu kadar büyük bir proje için üretici firmayla görüşüp ne kadarlık bir işlemciye ihtiyaç duyulacağının konuşulması ve karara bağlanması CPU ile ilgili beklenmedik durumları minimize eder.

Bu örnekler çoğaltılabilir ancak belirtilmek istene şudur; hangi kapasitede ne kadar işlemciye ihtiyaç duyulacağını önceden belirlemek oldukça göreceli bir durumdur. Ancak ihtiyaç olunacağı düşünülenden daha büyük kapasitelerde işlemciler almak ilerisi için iyi olacaktır. Çünkü SQL Serverların zaman geçtikçe daha yüksek performanslara ihtiyaç duyacağı çok az istisnası olan bir gerçektir.

#### 3.2.1.2. İşlemci hızı

İşlemci sayısını belirlemek gibi işlemcinin hızını belirlemek de senaryodan senaryoya değişebilen bir konudur. Bunun içinde eğer şartlar el veriyorsa alınabilecek en hızlı işlemciyi almak en az sıkıntıyla karşılaşılmasına ya da sıkıntılarla mümkün olduğunca geç karşılaşılmasına neden olur.

#### 3.2.1.3. CPU L2 önbelleği

İşlemcilerle ilgili tartışılan belki de en popüler soru; 2 tane küçük L2 önbelleğine sahip daha ucuz bir işlemci mi satın almalı yoksa büyük bir L2 önbelleğine sahip olan bir XEON işlemci mi almalı? Soruyu zor kılan nokta; L2 ön belleğine harcanacak paradan biraz kısıp daha küçük ön bellekli ancak daha hızlı bir işlemci alınabileceği gerçeğidir. Bu konuda şu şekilde bir kurallar dizisi koyulabilir:

Eğer sadece 1 ya da 2 işlemci alınacaksa alınabilecek en hızlı işlemci alınmalıdır . Bu durumda L2 önbelleği ikincil olarak düşünülebilinir. Ama eğer o konuda da bir seçim şansı var ise alınabilecek en büyük ön belleğe sahip işlemci alınmalıdır.Ama eğer 4 işlemci ya da daha fazlası satın alınabilecek durumdaysa L2 ön belleği maksimum tutularak işlemcilerin hızı ikincil plana itilebilir

### 3.2.2. Bellek

Bellek yönetimi SQL Server performansı için, işlemciden sonra ele alınıyor olmasına rağmen en az işlemci kadar önemlidir. Aslında bellek SQL Serverın performansını etkileyen en önemli donanımsal cihazdır. SQL Serverın bellek yönetiminden bahsederken, bahsedilen şey yalnızca fiziksel RAM'dir. SQL Server sanal belleği kullanmak için tasarlanmış bir server değildir.

İşletim sisteminin yaptığı gibi hem sanal belleği hem fiziksel belleği kullanan bir yaklaşım yerine SQL Server yalnızca fiziksel belleği kullanır. Bunun direkt sebebi de hız kriteridir. RAM içindeki veriye erişim onu diskten almaktan daha hızlıdır. Keza verinin yazılması içinde aynı şeyi söylemek mümkündür.

SQL Server kullanmak istediği belleği RAM'de bulamazsa sanal belleğe değil disklere yönelir. SQL Serverın önbellekleme mekanizması işlerim sistemininkine göre daha hızlıdır.

SQL Server için ayrılmış olan belleğin yeterli olup olmadığına karar vermenin en kolay yöntemi Buffer Cache Hit Ratio sayacını kontrol etmektir. Eğer bu sayacın değerleri %99 veya üzerindeyse yeterli belleğe sahiptir sonucu çıkarılabilir. %90 ile %99

arasındaysa ve SQL Server performansı mutluluk verici seviyelerdeyse bellek yine yeterlidir denebilir. Ama eğer bu seviye %90'ın altındaysa SQL Server performansı bundan kesinlikle olumsuz bir şekilde etkilenir. Tabi server OLTP için değil de OLAP için kullanılıyorsa %90'ın altında ama bu değere yakın sonuçlarda yeterli ram göstergesidir.

Bu işin de ideal olanı; bir SQL Server'daki fiziksel RAM miktarı o Server'ın en büyük veritabanından daha büyük olmalıdır. Örneğin 1 GB2lık veriye sahip olan bir veritabanını üzerinde barındıran bir SQL Server için en az 1GB'lık fiziksel RAM'e ihtiyaç vardır.

SQL Server %99'lık bir tampon önbellek erişim seviyesi ile oldukça hızlı çalışabilir. Bunun anlamı ihtiyaç duyulduğu zamanların %99'unda SQL Server ihtiyaç duyduğu veriyi tampon önbellekten almıştır. Sonuç itibariyle tampon önbellek erişim seviyesi %90'ın altındaysa acilen sisteme RAM eklenmesi, bellek yetmezliklerinden kaynaklanabilecek performans düşmelerine engel olacaktır.

### 3.2.3. Disk Depolama

Bellekten sonra disk konusu SQL Serverın performansına etki anlamında en önemli donanım cihazıdır. Aynı zamanda da karışık bir konudur. Bu bölümde disk performansının maksimize edilebileceği noktalardan bahsedilecektir.

### 3.2.3.1. Server'daki kullanılabilir boş disk miktarı

Performansa etkisi çok büyük olmasa bile disk dizisindeki bütün disklerin toplam alanlarının en azından %20'sinin boş olması önerilen eşik değeridir. Bunun sebebi ise yeni nesil Windows serverların hemen hepsinde kullanılması önerilen ve nerdeyse mecbur olan NTFS dosyalama sisteminin gerçek performansında çalışabilmesi için boş disk alanını beklemesidir.

Eğer SQL Server'ı barındıran makinenin disklerinde en az toplam boyutunun %20si kadar boş alan yok ise; bir şekilde dikte boş alan oluşturulmaya çalışılabilir. Diskten gereksiz dosyalar silinebilir (geri dönüşüm kutusu boşaltılabilir, geçici dosyalar temizlenebilir, kurulum dosyaları kaldırılabilir vs.) . Ya da Ek diskler alınarak disk dizisine eklenebilir.

### 3.2.3.2. Her bir dizideki fiziksel disk sayısı

Disk dizisi denen mekanizma iki ya da daha fazla diskin tek bir birimmiş gibi bağlanıp çalışmasıyla oluşan mekanizmadır. Örneğin RAID 5 sistemi için 4 adet fiziksel disk bulunmalıdır. Peki ama her bir disk dizisinde kaç adet disk bulunduğunu bilmenin SQL Serverın performansı açısından ne gibi bir yararı vardır?

Örneğin RAID 5 dizisine sahip olan bir disk alınacak ve en az 100 MB'lık müsait alan olacak. Üretici firmanın da 2 farklı seçenek sunduğu varsayılsın: 4 - 36GB drives (108GB müsait) ve 7 - 18GB drives (108GB müsait) .

Her ikisi de 100 MB'lık boş alan beklentisine cevap vermektedir. Peki ama hangi disk daha iyi yazma ve okuma performansı gösterecektir? Cevap ikinci seçenek olan 7–18 GB'lık disktir. Sebebi ise; en basit düşünce olarak disk dizisinde daha fazla disk olması demek aslında verileri okuyan disk başı sayısının da fazla olması demektir. Örneğin SCSI diskler veriyi aynı anda okuyup yazabilme yeteneğine sahiptirler. Dolayısıyla ne kadar fazla disk varsa disk dizisinde o kadar hızlı okuma ve yazmadan söz edilebilir. Dizideki her bir disk iş yükünün bir kısmını alır ve bu da performans için tartışmasız daha iyi bir yaklaşımdır. Disk kontrolcüsüne göre disk sayısının da belirli limitleri vardır ama genel olarak ne kadar fazla disk var ise performans açısından o kadar iyidir.

Yapılan denetim sonucunda veritabanı verilerinin tutulması için 2 tane disk dizisine sahip olunduğu belirlenmiş olsun . Ve ger biri 18'er GB'lık 3 er diskten oluşan RAID 5 disk dizileri olsun . Bu durumda bu 2 disk dizisine tek diziye çevirip 6 tane 18 GB'lık tek dizide birleştirmek daha hızlı I/O performansı oluşturacaktır.

#### 3.2.3.3. SQL Server için kullanılan RAID seviyesi

Her bir RAID seviyesinin kendisine göre artıları ve eksileri vardır:

RAID 1: İdeal olarak işletim sistemi ve SQL Serverın çalıştırılabilir dosyaları RAID 1 disk dizisinde yer almalıdır.SQL Serverın veritabanları eğer çok küçükse ve hepsi tek bir diske sığabilecekse bütün hepsini RAID 1'de depolamayı düşünmek en iyi performansı gösterir.Eğer mümkünse her bir işlem kayıt dosyasının kendine ait bir RAID 1 dizisinde tutulması birbirlerinin I/O işlemleriyle çakışması engelleneceği için oldukça iyi bir performans yükseltme çözümüdür.

RAID 5: En popüler RAID dizisi RAID 5 olmasına karşın en iyi SQL Server I/O performansını sağlayan dizi RAID 5 değildir. Eğer bir veri tabanındaki işlemlerin %10'dan fazlası yazma işlemi ise özellikle bu tarz veritabanları RAID 5'in dezavantajlarını hissedecektir. RAID 5 yalnızca okunabilir özelliklere sahip olan veritabanları için düşünülebilir. Microsoft'un yaptığı testler sonucunda çıkan sonuca göre RAID 5, RAID 10'a göre en fazla %50 daha yavaş olabilir.

RAID 10: RAID 10, en pahalı RAID çözümüdür ancak SQL Server için performansı en yüksek olan RAID dizisi çözümüdür. Yazım işleminin daha çok olduğu veri tabanlarında RAID 10 performansını daha iyi hissettirir.

RAID 10 işlem kayıtlarının tutulması ve gerektiğinde okunması içinde oldukça iyi bir çözümdür.

#### 3.2.3.4. Donanım yazılım RAID karşılaştırılması

RAID donanımsal olarak uygulanabildiği gibi işletim sistemi üzerinden yazılımsal olarak da uygulanabilen bir yöntemdir. Yazılımsal RAID kullanımı hiç bir zaman düşünülmemesi gereken bir çözümdür. Donanımsal RAID'e göre kat be kat düşük performansı vardır. Her zaman donanımsal RAID kullanılmalıdır.

#### 3.2.3.5. Disk bölünme seviyesi

Eğer veritabanı yeni bir disk dizisi üzerinde oluşturuluyorsa bütün dosyalar birbirini takip eden sayfalara yazılacaktır. Veritabanı üzerinde veri yazımı veri silinmesi gibi işlemler oldukça disk üzerinde bölünmeler oluşacaktır. Dosya bölünmesi, dosyaların

disk üzerinde boş bulunan yerlere yazılmasına sebep olur bu da okumayı oldukça güçleştirir.

Performans denetimin bir parçası olarak, veritabanının üzerinde koştuğu serverın ne kadarda bir birleştirme işlemine tabi tutulduğu öğrenilmelidir. Windows 2000 veya 2003 üzeriden disk birleştirmesi için kendi üzerlerinde gelen bir sistem aracı bulunmaktadır.

Eğer analizler sonucunda diski birleştirme yapma gerekliliği çıktıysa mümkün olan en yakın zamanda yapılmalıdır. Çünkü bir SQL Serverın veritabanı ve işlem kayıt dosyalarının birleştirilmesi kolay bir iş değildir. Örneğin SQL Serverın MDF ve LDF dosyaları Windows Server 2000 ve sonrasında gelen sürümlerdeki disk birleştirme aracıyla birleştirilemezler. Bu sebepten dolayı Diskeeper 8.0 gibi programlar ile belirli zaman aralıklarında SQL Server kapatılıp birleştirme işlemi yapılmalıdır. Bu işlem de veri tabanının büyüklüğüne, ne kadar parçalanma işlemi olduğuna göre uzayabilir.

Eğer gerçekten bir veritabanı performans sorunu yaşanıyorsa birleştirme işlemi oldukça ucuz olan bir yöntemdir. Sadece bazı durumlarda zamansal olarak uzun sürüyor olabilmesi tek dezavantajıdır. Ama bu işin ideali belirli zaman aralıklarında veri tabanına ve işlem kayıtları dosyalarına bu birleştirme işleminin uygulanmasıdır. Bu şekilde bu çok sık rastlanan I/O performansı sıkıntısının önüne geçilmiş olunur.

#### 3.2.3.6. İşletim sisteminin yeri

En iyi performans için işletim sistemi dosyalarıyla SQL Server veri dosyaları aynı disk üzerinde ya da en azından aynı sürücü üzerinde bulunmamalıdır.

### 3.2.3.7. SQL Server çalıştırılabilir dosyalarının yeri

SQL Server çalıştırılabilir dosyalarının (programı oluşturan exe'ler, dll'ler ini file'lar vs…) veritabanı veri dosyalarıyla aynı dizinde aynı sürücüde ya da yanı diskte bulunması çok kritik öneme sahip olan bir seçim değildir. Sadece genel yaklaşım olarak SQL Server veri dosyalarıyla aynı disk dizisinde yer almaması önerilir.

### 3.2.3.8. Takas dosyalarının yeri

SQL Serverın kurulu olduğu makinenin, SQL Server'a adanmış olan bir makine olduğu varsayılırsa ve de belleğin varsayılan ayarıyla bırakılıp dinamik özelliğine ayarlandığı varsayılırsa takas dosyalarının birçok aktiviteyi görmeyeceği söylenebilir. Çünkü SQL Server takas dosyalarını çok fazla kullanmaz. Bu yüzden de takas dosyalarının nerede olduğunun çok bir önemi yoktur. Sadece genel yaklaşım olarak SQL Server veri dosyalarıyla aynı disk dizisinde yer almaması önerilir.

Ama eğer SQL Serverın kurulu olduğu makine ona adanmış bir makine değilse ve sayfalama işlemlerinin sistemde görülmesi olası ise takas dosyalarının yeri sadece onlar için ayarlanmış olan bir disk dizisi olmalıdır. Ama hiç bunu düşünmeden mümkünse SQL Server makinesi sadece veritabanı işlemlerine adanmış olan bir makineye kurulmalıdır.

### 3.2.3.9. Tempdb veritabanının yeri

Eğer tempdb veritabanı yoğun olarak kullanılıyorsa, kendisine ait bir RAID1,RAID10 dizisine veya kendisine ait bir sürücüde barındırmak, performansı arttırmak açısından

iyi olacaktır. TempDb'ye yazılma ihtimaline karşın RAID-5'in kullanılmasından kaçınılmalıdır. Eğer kendine ait bir disk dizisi ayarlanamıyorsa o zaman da veritabanını, veri dosyalarıyla aynı diziye koymaktansa işletim sistemi dosyalarıyla aynı diziye koyulması tercih edilmelidir.

Eğer tempdb veritabanı çok fazla kullanılıyorsa ve varsayılan boyutu gittikçe büyüyorsa bu veritabanı için bir otomatik büyüme miktarı belirlenmelidir. Çünkü SQL Server servisi olan ms sqlserver her başladığında tempdb veritabanı varsayılan büyüklüğüyle tekrar yaratılmaktadır. Bu veritabanı büyürken bunu yapmak için kaynaklardan kullanır. Eğer SQL Server başlatılırken tempdb veritabanının büyüklüğünü doğru bir şekilde ayarlamak mümkün olursa, büyüme işlemi sırasında gerçekleşecek olan kaynak kullanımından kurtulunmuş olunur.

Bunlara ek olarak aslında tempdb veritabanı üzerindeki yoğun aktiviteler uygulamanın performansını düşürmektedir. Özellikle birden fazla geçici tabloyu oluşturup bunları bir de birleştirmek bu performansı iyice aşağıya çeker. Bu sıradaki sorgu hızlarını yüksek tutabilmek için AUTOSTATS veritabanı özelliğinin tempdb veritabanı için açık durumda olduğundan emin olunmalıdır.

### 3.2.3.10. Sistem veritabanlarının yerleri

Sistem veritabanları (master, msdb, model) çok fazla okuma ve yazma işlemine tabi olmayacağı için SQL Server veritabanı veri dosyalarıyla aynı diskte bulunmasının herhangi bir negatif etkisi olmayacaktır. Bunun sadece bir tek istisnası vardır. O da binlerce kullanıcısı olan çok büyük veritabanlarıdır. Bu durumlarda da sistem veritabanlarını kendi disk dizilerine koymak performans açısından pozitif bir yaklaşım olacaktır.

#### 3.2.3.11. Kullanıcı veritabanlarının yerleri

En iyi performans için kullanıcı veritabanı dosyaları yani MDB dosyaları kendi disk dizinlerinde yer almalıdır. Kayıt dosyaları olan MDL'lerden de ayırmak gerekmektedir. Eğer bir SQL Server üzerinde birden fazla geniş veritabanı var ise onların her birinin MDB'lerini de ayrı ayrı disklere koymak gerekmektedir.

#### 3.2.3.12. Kayıt dosyalarının yerleri

İdeal olarak her bir kayıt dosyası kendine ait bir disk dizisinde olmalıdır. Çünkü genellikle işlem kayıtları ardışık bir şekilde yazılır ve eğer dizi kayıtları ardışık bir şekilde yazabilecek durumdaysa (yani diğer dosyaların yazma veya okuma isteklerinden dolayı bu yazma işlemi bir kesintiye uğramıyorsa) bu ardışık yazma işlemi oldukça hızlı olacaktır. Ama eğer bu ardışık yazma işlemi aynı anda gelen yazma veya okuma taleplerinden dolayı gerçekleştirilemiyorsa ardışık yazma işlemi gerçekleşemez ve hem okuma sırasında hem de yazma sırasında performans sıkıntıları yaşanır.

Teoride doğru olan her bir kayıt dosyası için ayrı bir disk dizisi almak iken pratiğe geçişte pahalı bir yaklaşımdır. Eğer bu durum karşılanmıyorsa baz alınması gereken nokta en azından kayıt dosyalarının ana veri tabanı veri dosyalarından ayrılmasıdır.

#### 3.2.3.13. Sunucudaki disk kontrolcülerinin sayısı

SCSI de olsa fibre de olsa her bir disk kontrolcüsünün bir limiti vardır. Bu yüzden sahip olunacak kontrolcü sayısının karşılayacağı veri miktarıyla beklenen veri miktarının birbiriyle örtüşmesi beklenir. Bu durumda yapılacak çok net bir şey yoktur ancak genel olarak yapılan önerme en azından 2 tane disk kontrolcüsüne sahip olunursa bunlardan

bir tanesi hard disk dışındaki depolama araçları için kullanılabilirken (CD-ROM,DVD-ROM,yedekleme cihazları vs.) bir diğeri hard disk için kullanılabilir. Buradaki amaç hızları birbirinden farklı olana depolama araçlarını aynı yerde toplamamak bunları gruplandırmaktır.

Bu önermeye göre genellikle görülen uygulama tipi şöyledir; hard-disk dışındaki depolama araçları için bir kontrolcü, RAID 1 dizisi için bir kontrolcü ayarlandıktan sonra bir ikinci kontrolcü de SQL veri tabanı kayıt dosyalarını ve veri dosyalarını tutmak için ayarlanabilir. Ama dikkat edilmesi gereken nokta bir disk kontrolcüsünün kontrol edebileceği maksimum sürücü sayısı belliyken aşırı yüklenmelerde performans sıkıntılarına sebebiyet verecektir.

#### 3.2.3.14. Sunucudaki disk kontrolcülerinin tipi

Maksimum SQL Server performansı için alınabilecek en hızlı disk kontrolcüsü alınmalıdır. Her bir kontrolcünün değişik tipleri olduğu bilinmektedir. Örneğin SCSI disk kontrolcüsü Geniş SCSI, Dar SCSI, Ultra SCSI gibi alt gruplara ayrılmaktadır. Fibre disk kontrolcüler içinde bu tarz çeşitlilikler mevcuttur. Buradaki seçim baz noktası çok karışık değildir. Eldeki bütçeye göre maksimum hıza sahip olan disk kontrolcüsü alınmalıdır.

#### 3.2.3.15. Sunucudaki disk kontrolcülerinin önbellek miktarı

Bir disk kontrolcüsünün ön bellek miktarı da sorgulanması gereken bir ölçüttür. Bazı disk kontrolcüler ekstra disk ön belleği eklenmesine müsaade ederler. SQL Server I/O performansı konusunda çok hassas olması ve bunun arttırtmasının SQL Server

performansına direkt etki etmesi sebebiyle ne kadar fazla ön belleğe sahip olan bir disk kontrolcüsü kullanılırsa o kadar iyi olur.

### 3.2.3.16. Disk kontrolcüsündeki önbelleğe geri yaz (write back cache) ayarının durumu

Disk kontrolcüsündeki önbellek hızı arttırma yolu olarak 2 seçenek sunar. Bir tanesi yazma için bir tanesi de okuma içindir. Diskin yazma hızıyla ilgili olan ön belleğe geri yazım ise SQL Server tarafından açık olmadığı varsayılan bir ayardır. Bu yüzden bu ayar eğer açık ise gerçekten kapatılmalıdır. Aksi takdirde bazı durumlarda veri yazımında kesilmeler oluşarak veri kaybına neden olunabilir. Çünkü bir defa veri yazıldı mı SQL Server o verinin yazıldığını kabul eder ancak elektrik enerjisi kaybı gibi bazı senaryolarda ön belleği geri yazma özelliği veriyi geri yazmaz.

Bu tarz durumların önüne geçmek için bazı disk kontrolcülerinin üzerlerinde enerji yedekleme üniteleri bulunmaktadır. Daha hızlı ama yanlış yazılabilecek verilerdense daha yavaş ama yazıldığı garanti olan verilerle çalışmak doğru bir seçimdir. Bu yüzden de doğru yaklaşıma uygun olarak yapılması gereken, Önbelleğe geri yaz özelliğinin kapatılmasıdır.

### 3.2.3.17. Disk sürücülerinin hızı

Disk dizileriyle gelen sürücüler farklı hızlar da olabilir. Burada da tercih sebebi çok açıktır. Hızlı olan sürücü tercih edilir. Eğer aynı disk dizisi içinde farklı hızlarda diskler kullanılırsa bu önemli sıkıntılara sebep olabilir. Bir diziye bir verinin bir kısmı çoktan yazılmışken aynı büyüklükteki bir sayfa diğer diske henüz yazılmamıştır. Diğer yazılan

sayfa da yeni veri yazması için yavaş olanı beklemek zorundadır. Bu tarz bir işlem yapılırsa performans sıkıntıları yaşanacağı açıktır.

#### 3.2.3.18. Sunucudaki ağ kartlarının miktarı

SQL Server için normal bir uygulamada 1 tane ağ kartı oldukça yeterlidir. Ancak uygulamanın yaptığı işe göre bu durum değişebilir. Örneğin binlerce kullanıcı bu SQL Server'a veri yazıyor ya da ağır veriler çekiyorsa ağ trafiği bundan yorulacaktır. Bu durumda performansı arttırmak için ek ağ kartlarını sisteme doğru bir şekilde dâhil etmek herhangi olumsuz bir sonuç doğurmayacaktır.

#### 3.2.3.19. Sunucudaki ağ kartlarının hızları

Sistemdeki kartlar en azından 100Mbs'lik hıza sahip olmalıdır. 10 MBs'lik bir ağ kartı SQL Server için yeterli olmayacaktır. Eğer birden fazla 100 MB'lık ağ kartları beklenen verimi sağlayamazsa gigabit kartlar düşünülmelidir. Daha hızlı ağ kartı ağ trafiğini hızlandırmayacaktır. Sadece bir t anında kabul edilebilecek trafik miktarı artacaktır.

#### 3.2.3.20. Ağ kartlarının anahtarla (switch) bağlantısı

Büyük veri merkezleri için geçerli olabilecek bu durum küçük organizasyonlar için HUB kullanılarak giderilebilir. Eğer mümkünse en azından switch kullanılarak tam dubleks modunda ve 100 Mbs'lik bir hızda çalışması sağlanmalıdır. Yalnız unutulmaması gereken birçok kritik noktalar vardır. SQL Server'ı kullanan uygulama kod anlamında, web server anlamında gelecek olan bu trafiğe hazır mıdır, SQL Server'ın kurulu olduğu

makine bu trafiğe yanıt verebilecek donanımsal yetiye sahip midir gibi sorular mutlaka düşünülmelidir.

#### 3.2.3.21. Donanımların en son güncellenmiş sürücü yazılımlarının kontrolü

Donanımların son güncel yazılımlara sahip olmaması gerçekten çok büyük karmaşıklıklara sebep olabilir. Belki yükseltilen bir sürücü yazılımı versiyonu o sürücünün daha hızlı çalışmasına ve böylece toplam performansın arttırılmasına sebep olabilecekken hem bu artıştan yoksun kalınabilir hem de güncellemenin onardığı bug'larla uğraşılmak zorunda kalınabilir. Bu yüzden sistemli bir şekilde donanımların son güncel yazılım versiyonlarına sahip olması sağlanmalıdır.

#### 3.2.3.22. Server'ın sadece veritabanı sunucusu için ayrılması

SQL Server kesinlikle sadece ona adanmış bir makineye kurulmalıdır. Aksi halde makineyi paylaştığı diğer uygulamalarla bütün kaynakları paylaşmak durumunda kalacaktır. Bu sebepten performans arttırmak için ne kadar ayar yapılırsa yapılsın istenilen sonuca ulaşılamayacaktır.

### 3.3. İşletim Sistemi Performans Kontrolü

Tablo 3.3. İşletim sistemi performans değerlendirme kontrol listesi

| İşletim Sistemi Performans Kriterleri | Değerler |
|---|---|
| Hangi işletim sistemi kullanılmakta? | |
| Disk bölümleri NTFS 5.0 kullanılarak mı formatlandı? | |

Tablo 3.3. İşletim sistemi performans değerlendirme kontrol listesi (Devam)

| | |
|---|---|
| NTFS data dosyasında “şifreleme ve sıkıştırma" özelliği kapalı mı? | |
| En son servis paketi yüklü mü? | |
| En son Microsoft-onaylı donanım sürücüleri yüklü mü? | |
| Windows server tek başına çalışmak için mi konfigüre edildi? | |
| “İşlemci Zamanlaması” seçeneği, arka planda çalışan servisler için performansı yükseltecek şekilde mi ayarlandı? | |
| Güvenlik denetimi açık mı? | |
| PAGEFILE.SYS takas dosyası ne kadar büyüklüktedir? | |
| Gereksiz servisler kapatıldı mı? | |

### 3.3.1. İşletim sistemi seçimi

SQL Server gibi Windows Server da kendini optimize eder. Ama SQL Server gibi, Windows’un da performansının artırabileceği noktalar vardır. Windows Server’ın performansını artırmak demek SQL Server’ın da performansını artırmak demektir.

SQL Server’da en iyi performansı alabilmek için Windows 2003 kullanılmalıdır, çünkü 2000 ve NT 4.0’a kıyasla birçok performans gelişimi özelliği sunar. Bunlardan bazıları: Intel’in hyper-threading özelliğine sahip işlemcilerinde daha iyi performans sunması. Intel chipler ile 32 CPU ve 64GB RAM, Itanium chipleri ile 64 CPU’ları ve 512GB seviyelerine ulaşan RAM desteklemesi. I/O yolu ve disk I/O performansı artırılmış, I/O isteklerini karşılamak için gerekli CPU zamanı azaltılmıştır.

### 3.3.2. Disk bölümleri için dosya sistemi seçimi

Eğer server yeniyse ve Windows 2000 ya da Windows 2003 yüklüyse, tüm sürücüler NTFS 5.0 ile formatlanmıştır. Eğer server eskiyse ve önceden Windows NT 4.0 Server yüklüyse ve sürücüler Windows 2000 ya da 2003'e yükselterek formatlanmadıysa, diskler muhtemelen NTFS 4.0 ile formatlanmıştır. NTFS 4.0 ile 5.0 arasında çok fark olmasa da yükseltme yapmaya değecektir. NTFS 5.0 dosya bulmak için daha az dosya erişimi ve genelde daha hızlı disk okuma gibi performans gelişimlerine sahiptir. Windows 2000 ve 2003'ten önce, bazı DBA'ler sürücüleri FAT olarak formatlardı, çünkü az da olsa NTFS 4.0'dan daha iyi performans sağlıyordu. NTFS 5.0'dan itibaren böyle bir durum bulunmuyor, bu nedenle tüm SQL Server kullanan tüm diskler en iyi performans için NTFS 5.0 ile formatlanmalıdır.

Eğer canlı uygulamadaki SQL Server Windows 2000 altında NTFS 4.0 olarak formatlanmış bölümler kullanıyorsa, onları NTFS 5.0'e dönüştürmek kolay olmayacaktır. Bu durum göz ardı edilebilir çünkü fayda maliyet optimizasyonu yapıldığı zaman çok da büyük bir fayda sağlanmayacaktır. Ama Windows NT 4.0 Server'dan Windows 2000'e yükseltme söz konusuysa disk kesinlikle NTFS 5.0 ile formatlanmalıdır.

### 3.3.3. NTFS veri dosyasında şifreleme ve sıkıştırma özelliğinin durumu

Windows 2000 altındaki NTFS 5.0, dosya şifreleme ve sıkıştırma destekler ve varsayılan seçenek olarak yeni yüklenmiş Windows 2000 veya 2003 Server'da bu özellik kapalıdır. Bu özellik sınırlı durumlar için faydalı olsa da SQL Server için herhangi bir fayda sağlamaz. Gerçekte bu özelliklerin bir ya da ikisini kullanmak performansı ciddi şekilde düşürür.

SQL Server I/O konusunda çok hassastır ve disk I/O'sunu artıran her şey SQL Server'ın performansını düşürür. Dosya şifrelemek ve sıkıştırmak, dosyalar kullanıldığı anda manipüle edilmelerini gerektirdiğinden disk I/O'sunu artırır. Bu nedenle sıkıştırmak da şifrelemek de eğer SQL Server dosyalarında kullanılıyorsa performans ciddi şekilde zarar görür.

### 3.3.4. Server'da yüklü olan servis paketinin sürümü

Tüm servis paketleri en az bir ya da daha fazla performans gelişimi özelliğine sahiptir. Bunun nedeni ayarlamanın Microsoft tarafından yapılıyor olması ya da performansı düşüren bir hatanın ortadan kaldırılmış olmasıdır.Microsoft tarafından dağıtılmaya başlandığı gün yüklemek zorunlu olmasa da servis paketlerini mümkün olduğunca çabuk yüklemek kesinlikle faydalı olacaktır.

### 3.3.5. Microsoft onaylı donanım sürücülerinin kontrolü

Sorunlu donanım sürücüleri Windows 2000 ve 2003'te bazı performans sorunlarına neden olur. Bunlar çoğunlukla disk ya da ağ kaynaklı sorunlardır. Periyodik olarak en son Microsoft-onaylı donanım sürücülerinin yüklü olup olmadığını kontrol edilmelidir. Bu işlemi donanım üreticisinin web sitesinden ya da Microsoft Update hizmetiyle yapılabilir. Bazı durumlarda üretici yeni sürücü yayınlar ancak Microsoft tarafından onaylanması biraz gecikir. Bu durumda sabırlı olup beklemekte yarar vardır. Performansı artırmak önemli olsa da yazılım kararlılığı çok daha önemlidir.

### 3.3.6. Windows server'ın çalışma tipinin belirlenmesi

Bir Windows 2000 ya da 2003 server tek başına çalışan bir sunucu ya da bir domain kontrolcüsü olarak konfigüre edilebilir. En iyi performans için SQL Server koştuğu makine tek başına çalışan server olarak konfigüre edilmelidir. Bunun nedeni bir domain kontrolcüsünün kaynaklarını SQL Server'dan başka işlemler için de ayırıyor olmasıdır, bu da performansı olumsuz etkiler.

### 3.3.7. İşlemci zamanlaması seçeneği ve arka planda çalışan servislerin performansla ilgili ayarları

Windows 2000'de, denetim masasında, sistem ikonu altındaki gelişmiş sekmesinde, performans seçenekleri seçildikten sonra işlemci zamanlaması özelliğini görüntülenip ayarlanabilir. En iyi SQL Server performansı için arka planda çalışan servislerin performansının arttırması seçilmelidir. Bu şekilde işletim sisteminin ön plandaki uygulamalar yerine SQL Server gibi arka planda çalışan uygulamalara öncelik tanıması sağlanır.

Ayrıca yine aynı yerde bellek kullanım özelliği ayarlanabilir. En iyi SQL Server performansı için programlar sekmesini seçilir ve buradan yapılan ayarla işletim sisteminin, sistem önbelleği yerine SQL Server gibi programlara daha fazla bellek ayırması sağlanmış olur.

### 3.3.8. Güvenlik denetiminin kontrol edilmesi

Windows 2000 ve 2003 server üzerindeki tüm aktiviteleri denetleme özelliğine sahiptir. Varsayılan değer olarak bu özellik kapalı gelir. En iyi performans için bu denetim açılmamalı ve başka denetim açılmamalıdır. Eğer açmak zorundaysa performansı etkilemesini asgariye indirebilmek için denetimi sınırlandırılmalıdır.

### 3.3.9. PageFile.sys takas dosyasının büyüklüğü

Microsoft pagefile.sys dosyasının RAM miktarının 1,5 katı olmasını önermektedir. İhtiyaç olan boyut çalışan ek SQL servislerine bağlıdır. Eğer "tam metin arama" çalışıyorsa Microsoft Pagefile.sys'nin RAM miktarının 3 katı olmasını öneriyor.

Microsoft'un önerileri iyi bir başlangıç noktasıdır. Ama PAGEFILE.SYS dosyasının boyutunu ayarlamanın en iyi yolu dosyaya ait performans izleme sayfasını kullanarak üretim sürecinde dosyanın ne kadarının kullanıldığını izlemektir. Sonrada minimum aktif olarak kullanılan kısmından biraz büyük, maksimum da minimum boyuttan 50 MB fazla olacak şekilde yeniden boyutlandırılmalıdır.

PAGEFILE.SYS düzenlemesi Windows 2000'de şu yoldan yapılabilir: Bilgisayarım;Özellikler;Gelişmiş;Performans Özellikleri;Sanal Bellek. Bu özellik değiştirildikten sonra bilgisayar yeniden başlatılmalı. Windows 2003'te izlenmesi gereken yol: Denetim Masası;Sistem;Gelişmiş;Performans; Ayarlar ;Gelişmiş ;Sanal Bellek

### 3.3.10. Gereksiz servislerin kontrolü

En iyi performans için tüm gereksiz servisler kapatılmalıdır. Bu işlem hem RAM hem de CPU performansını artırır. Aşağıda işletim sisteminin çalıştırdığı ve gereksiz olarak nitelenen servislerden bir kısmı yer almaktadır. Bazıları Server'da yüklü olmayabilir, bazıları da varsayılan değer olarak kapalı olabilir.

Alerter, Application Management, Clipbook, Distributed Link Tracking Server, Fax Service, File Replication, FTP Service, Indeksing Service, Internet Connection Sharing, Intersite Messaging, Kerberos Key Distribution Center, License Logging Service, Logical Disk Manager Administrative Service, Messenger, Microsoft Search, NetMeeting Remote Desktop Sharing, Network DDE, Network DDE DSDM, Print Spooler Service, QoS RSVP, Remote Access Auto Connection Manager, Remote Procedure Call (RPC) Locator, Routing and Remote Access, RunAsService, Smart Card, Smart Card Helper, SMTP Service, Telnet, Utility Manager, Windows Installer, World Wide Web Service.

## 3.4. SQL Server Konfigürasyonu Performans Kontrolü

Tablo 3.4. SQL Server konfigürasyonu performans kontrol listesi

| **SQL Server Konfigürasyonu** | **Gelişmiş Ayar?** | **Yeniden Başlat?** | **Varsayılan Değer** | **Mevcut Değer** |
|---|---|---|---|---|
| affinity mask | Evet | Evet | 0 | |
| awe enabled | Evet | Evet | 0 | |
| cost threshold for parallelism | Evet | Hayır | 5 | |

Tablo 3.4. SQL Server konfigürasyonu performans kontrol listesi (Devam)

| | | | | |
|---|---|---|---|---|
| cursor threshold | Evet | Hayır | -1 | |
| fill factor (%) | Evet | Evet | 0 | |
| indeks create memory (KB) | Evet | Hayır | 0 | |
| lightweight pooling | Evet | Evet | 0 | |
| Locks | Evet | Evet | 0 | |
| max degree of parallelism | Evet | Hayır | 0 | |
| max server memory (MB) | Evet | Hayır | 2147483647 | |
| max text repl size (B) | Hayır | Hayır | 65536 | |
| max worker threads | Evet | Evet | 255 | |
| min memory per query (KB) | Evet | Hayır | 1024 | |
| min server memory (MB) | Evet | Hayır | 0 | |
| nested triggers | Hayır | Hayır | 1 | |
| Network packet size (B) | Evet | Hayır | 4096 | |
| open objects | Evet | Evet | 0 | |
| priority boost | Evet | Evet | 0 | |
| query governor cost limit | Evet | Hayır | 0 | |
| query wait (s) | Evet | Hayır | -1 | |
| Recovery interval (m) | Evet | Hayır | 0 | |
| scan for startup procs | Evet | Hayır | 0 | |
| set working set size | Evet | Evet | 0 | |
| user connections | Evet | Evet | 0 | |

Bu bölümde bazı performans bağlantılı SQL Server ayarları incelenecektir. Bu işlemler Enterprise Manager ya da “sp_configure” ile yapılır.

Eğer ilk defa SQL Server ile çalışılıyorsa ilk olarak, değişik konfigürasyon ayarları kontrol edilmeli ve varsayılan değerlerden hangilerinin farklı olduğunu görebilmek için karşılaştırma yapılmalıdır. Değişiklik olan ayarlar fark edildikten sonra değişikliğin nedenini öğrenilmelidir. Eğer neden anlaşılamıyorsa ya da bulunan neden önemsizse, varsayılan değer geri yüklenmelidir.

SQL Server 2000 üzerinde 36 farklı ayar seçeneğinden en önemli olan 23 tanesi bu bölümde ele alınmıştır.

SQL Server konfigürasyon ayarlarını incelemenin en kolay yolu Query Analyzer'da "sp_configure" komutunu çalıştırmaktır Çalıştırılan bu komut sonunda ortaya çıkacak olan "name" kolonu SQL Server konfigürasyonunun adıdır. "minimum" ve "maximum" kolonları bu seçenek için girilebilecek değer aralığını gösterir. "config_value," konfigürasyon değerini gösterir. (bazı değerler SQL Server yeniden başlatılmadan ya da "reconfigure with override" seçeneği çalıştırılmadan aktif olmayacağı için SQL Serverın şu an kullandığı değeri göstermiyor olabilir). "run_value," şu an aktif olan değerdir. Eğer SQL Serverın en son çalıştığı andan itibaren bir değişiklik yapılmazsa son iki kolon aynıdır.

Bu konfigürasyon ayarları Enterprise Manager kullanılarak değiştirilir. Ama değişiklik yapmanın en kolay yolu SP_CONFIGURE komutudur. Bu komut şu şekilde çalıştırılır:

```
Sp_configure ['configuration name'], [configuration setting value]
GO
reconfigure with override
GO
```

Komut içerisindeki “configuration name, değişiklik yapılacak ayarı, “configuration setting value” ise ayarın nümerik değerini gösterir.

Sp_configure çalıştıktan sonra bir işlem daha yapılmalıdır. “Reconfigure” seçeneği (normal ayarlar) ya da “reconfigure with override” seçeneği kullanılmalıdır. Eğer kullanılmazsa değişiklikler kaydedilmez. Her seferinde “reconfigure with override” kullanmak daha kullanışlıdır. Eğer değişiklikler Enterprise Manager’dan yapılıyorsa “reconfigure with override” otomatik olarak çalıştırılır. Bir kez bunu yaptıktan sonra değişikliklerin çoğu hemen aktif olur. Aktif olmayanlar için SQL Server servisi durdurulup yeniden başlatılmalıdır.

### 3.4.1. Çoklu işlemlerin yürütülmesi (affinity mask)

SQL Server çoklu işlemler yürütebilmektedir ve bu durum genelde Server’daki işlemciler üzerindeki yükü dengelemede iyi sonuç verir. Bu işlemin tek olumsuz yanı, bir işlem bir işlemciden diğerine transfer edildiğinde, işlemci önbelleği yeniden yüklenmelidir ve bu da bazı durumlarda performansı olumsuz etkiler.

Dörtten fazla işlemciye sahip sistemlerde, performans her bir işlemciye belirli bir işlem atayarak artırılabilir. Bu sayede işlemci önbelleklerinin yeniden yüklenmesi engellenir. Örneğin SQL Server’ın sadece belirli CPU’ları kullanması sağlanabilir.

"Affinity mask" ayarının varsayılan seçeneği olan "0" SQL Server’ın bir işlemin yakınlığını(affinity) belirlemek için Windows Scheduling algoritmasını sağlar. Bir diğer deyişle hangi işlemin hangi CPU’da çalışacağına SQL Server değil, Windows karar verir. 4 veya daha az işlemcili sistemlerde varsayılan değer en iyi seçimdir.

Dörtten fazla işlemcili sistemler için daha uygun bir değer kullanmak için varsayılan değer değiştirilmek istenebilir. “Affinity mask” kullanarak CPU kullanımı sınırlamadan önce SQL Server’ın server üzerinde çalışan tek uygulama olduğundan emin olunmalıdır. Aksi takdirde değişiklik işe yaramaz.

Örneğin 8 işlemcili bir sistem varsa ve SQL Server 4 işlemciyi kullanmak üzere ayarlandıysa, 4 işlemci SQL Server için kullanılacağından önbellek yeniden yüklemeleri için zaman kaybedilmeyecek, geri kalan 4 işlemciyi de diğer servisler ve uygulamalar (IIS gibi) kullanacaktır. Böylece server uygulamalarının da performansı artmış olur.

### 3.4.2. Gelişmiş Windows uzantılarının kullanılması (AWE enabled)

Windows 2000 ya da 2003 altında SQL Server 2000 Standard Edition veya SQL Server 2000 Enterprise Edition kullanılıyorsa, ya da serverda 4GB’tan az RAM varsa bu ayar her zaman varsayılan değer olan 0’da kalmalıdır, bu da AWE hafızasının kullanılmadığını gösterir.

AWE (Advanced Windowing Extensions) API, AWE API kullanmak üzere tasarlanmış uygulamalara Windows 2000, 2003 Advanced Server ya da Datacenter Server altında 4GB’tan fazla RAM ile çalışma izni verir. SQL Server 2000 Enterprise Edition (SQL Server 2000 Standard Edition değil) AWE kullanabilir ve 4GB’tan fazla RAM kullanabilir. Eğer işletim sistemi Windows 2000 ya da 2003 Advanced Server, SQL Server 2000 Enterprise Edition 8GB’a kadar RAM kullanabilir. Eğer işletim sistemi Windows 2000 ya da 2003 Datacenter Server ise SQL Server 2000 Enterprise 64GB’a kadar RAM kullanabilir.

Server'da 4GB'tan fazla RAM varsa Windows 2000 ve 2003'ün (Advanced ve Datacenter), SQL Server 2000 Enterprise Edition ile birlikte 4GB'tan fazlasını kullanabilmesi için iki işlem yapılmalıdır.

AWE bellek desteğini konfigüre edebilmeniz server'ın sahip olduğu RAM miktarına bağlıdır. Öncelikle Windows 2000 ya da 2003 (Advanced ya da Datacenter) konfigüre edebilmek için, aşağıdakilerden birisi boot.ini dosyasına girilmelidir:

4GB RAM: /3GB (AWE desteği kullanılmıyor) , 8GB RAM: /3GB /PAE , 16GB RAM: /3GB /PAE , 16GB + RAM: /PAE

/3GB anahtarı işletim sisteminin SQL Server'a 4GB'tan 3ünü kullanmasına izin vermesini sağlar. Eğer bu seçenek belirtilmezse, SQL Server sadece 2GB kullanır.

AWE bellek teknolojisi sadece 4GB'ı geçen RAM için kullanılabilir, bu nedenle /3GB anahtarı Server'da kullanılabilecek maksimum miktarı belirlemek için gereklidir. Eğer sistemde 16GB'tan daha fazla RAM varsa /3GB anahtarı kullanılmalıdır. Bu durumda sistem AWE hafızasını kullanmak için 1GB'tan fazla alana ihtiyaç duyar.

İkinci adım, "awe enabled" ayarını 1'e getirmek ve SQL Server servisini yeniden başlatmaktır. Bu adımdan sonra SQL Server ekstra RAM'i kullanabilir.

"Awe enabled" ayarı değiştikten sonra dikkat edilmesi gereken nokta, artık SQL Server'ın hafızayı dinamik olarak kontrol etmeyeceğidir. Bunun yerine kullanılabilir tüm RAM'i alır (128MB'ını işletim sistemine bırakarak). Eğer SQL Server'ın tüm

mevcut RAM'in kullanılması önlenmek isteniyorsa "max server memory" ayarını kullanmalıdır.

Denetim işleminin bir parçası olarak, bu ayarın ne olduğunu kontrol edilmeli ve server donanım/yazılımına uygunluğunu kontrol edilmelidir.

### 3.4.3. Paralellik için maliyet eşik değeri (Cost threshold for parellesim)

Bir SQL Server sorgusu çalıştırmak için paralellik kullanmak maliyetlidir. Bunun nedeni sorguyu seri olarak çalıştırmak yerine paralel çalıştırmanın ek yük getirmesidir. Ancak paralel sorgu kullanımının yararları maliyetinden fazla ise, kullanılmalıdır.

Pratik bir kural olarak, eğer bir sorgu seri olarak çok hızlı çalışıyorsa, paralel çalıştırmayı düşünmek bile gereksizdir. Çünkü paralel çalıştırmanın getirdiği ek süre nedeniyle seri olarak çalıştığından daha yavaş çalışacaktır.

Varsayılan seçenekte, eğer Query Analyzer sorgunun 5 saniyeden az süreceğini hesaplarsa paralellik düşünülmez. İşte bu 5 saniye SQL Server'ın paralellik eşiğidir. Bu değeri 0'da 32767'ye kadar bir aralıkta değiştirebilirsiniz. Yani, 10 yaptığınızda, bir sorgunun çalışma süresi 10 saniyeyi geçmediği sürece SQL Server paralel çalıştırmayı denemeyecektir.

Çoğu durumda bu ayar değiştirilmemelidir. Ancak, eğer SQL Server çok fazla paralel sorgu çalıştırıyorsa ve CPU kullanım oranı yüksekse, bu değeri 5ten yüksek yapmak, paralel sorguların sayısını ve CPU kullanımını azaltarak genel performansı yükseltecektir.

Bu değeri 5'ten az girmek, çoğu durumda performansı düşürür. Daha düşük değerin işe yaradığı bir durum, veri ambarı gibi kullanılan ve çok kompleks sorguların çalıştırıldığı SQL Server'lardır. Düşük bir değer Query Optimizer'ın paralelliği daha sık kullanmasını sağlar. Eğer SQL Server'ın tek CPU'ya erişimi varsa, ya da "affinity mask" ayarı yüzünden tek CPU'ya erişebiliyorsa paralellik sorgu çalıştırılırken düşünülmez.

### 3.4.4. İmleç eşik değeri (Cursor Threshold)

Eğer SQL Server imleç kullanmıyor ya da nadiren kullanıyorsa, bu değer varsayılan değeri olan "-1"de kalmalıdır. -1 değeri SQL Server'ın tüm imleçleri senkronize olarak çalıştırmasını sağlar ve bu değer eğer Server'daki imleçler çok büyük sonuç kümeleri getirmiyorsa ideal değerdir. Eğer SQL Server'da çalışan imleçlerin çoğu ya da tümü, çok büyük sonuç setleri dönüyorsa imleçleri senkron olarak çalıştırmak faydalı bir işlem olmayacaktır.

"Cursor threshold" ayarı, büyük cursorlar için 2 farklı değer alabilir. "0" değeri, SQL Server'ın tüm imleçleri asenkron olarak çalıştırmasını sağlar ve çoğu imleçlerin büyük sonuç setleri getirdiğinde işe yarar.

Bazı imleçlerin sonuç setlerinin küçük, bazılarının büyük olduğu durumlarda yapılması gereken büyük ve küçük kavramlarını tanımlamak, bu kavramlara göre eşiği belirlemektir. Örneğin, 1000 kayıttan az sonuç getiren imleç küçük, kalanları büyük olarak tanımlandıysa, "cursor threshold" değeri 1000 yapılmalıdır.

"Cursor threshold" 1000 yapıldığında, 1000'den az kayıt getiren imleçler senkron, diğerleri de asenkron olarak çalıştırılır.

Çoğu durumda bu seçenek çok iyi çalışır. Tek sorun ideal eşiği belirleyebilmektir. Bunun için de denemek gerekir. Burada da en iyi değer varsayılan değerdir ve bu değeri sadece çok büyük imleçler kullanılıyorsa ve sonuçları test ediliyorsa değiştirilmelidir.

Denetimin bir parçası olarak, imleçlerin hangi sıklıkta kullanıldığı ve sonuç kümlerinin boyutu araştırmalı, en iyi ayar buna uygun olarak yapılmalıdır. Ayrıca imleç kullanımını kısıtlama yoluna da gidilebilir.

### 3.4.5. Doluluk oranı (Fill factor)

Bu seçenek indeksler oluşturulduğunda tanımlanan doluluk faktörü değerinin değiştirebilmesini sağlar. Varsayılan değer "0"dır. Bu da indeks sayfalarının %100 doldurulacağını gösterir, ancak ara sayfalar tamamen dolmaz. Girilebilecek değer aralığı 0-100'dür.

Varsayılan değer sadece, indeksleri tanımlarken belirli bir doluluk faktörü tanımlanmadığında devreye girer. Eğer indeksi oluştururken bir doluluk faktörü tanımlanırsa, varsayılan değer değil, belirlenen değer kullanılır.

Çoğu durumda, varsayılan değere dokunmamak ve değiştirilecekse de indeksi oluştururken tanımlamak en iyi yoldur. Doluluk faktörü değerinin 0'dan farklı olup olmadığı kontrol edilmelidir. Eğer farklıysa nedeni araştırılmalıdır. Eğer bulanamazsa ya da bulunan neden mantıklı değilse varsayılan değere dönülmelidir. Ayrıca değişiklikten sonra oluşturulan tüm indekslerin de yeni fill factor değeriyle oluşturulduğu, geri alabilmek için bu indeksleri yeniden oluşturulması gerektiğini unutulmamalıdır.

### 3.4.6. İndeks oluşumunda bellek kullanımı (Indeks create memory)

Bu ayar SQL Server'ın indeks oluşturma sıralamalarında ne kadar bellek kullanması gerektiğini belirler. Varsayılan değer "0" SQL Server'ın ideal değeri otomatik olarak belirlemesini sağlar. Çoğu durumda SQL Server en iyi değeri otomatik olarak belirler.

Çok büyük tablolar gibi sıra dışı durumlarda, SQL Server'ın hata yapması, geniş indekslerin çok yavaş oluşmasına ya da hiç oluşmamasına neden olması mümkün olabilir. Eğer bu durumla karşılaşılırsa, Indeks Create Memory ayarı değiştirilmek istenebilir. Girebilecek değer aralığı 704 – 2147483647'dir. Bu sayılar KB cinsinden RAM miktarını gösterir.

Eğer bu değişiklik yapılırsa, indeks oluşturmak için harcanacak belleğin başka bir işlemde kullanılamayacağı unutulmamalıdır. Eğer yeteri kadar bellek varsa sorun olmaz. Ancak yeterli RAM yoksa, bu değeri değiştirmek performansı olumsuz etkileyecektir. Bu değeri değiştirmeyi sadece çok büyük indeksler oluşturulurken değiştirmek düşünülebilir ve sonra değişiklik geri alınabilir.

### 3.4.7. Kilitler (Locks)

SQL Server'ın bir kaydı kilitlediği her durumda, kilit bellekte saklanır. Varsayılan değer olan 0 kilit belleğinin SQL Server tarafından dinamik olarak yönetildiğini gösterir. SQL Server kullanılabilir belleğin %2 - %40'ını kilitler için ayrılabilir. Ek olarak, eğer SQL Server, kilitler için ek alan ayırmanın işletim sistemi seviyesinde paging'e neden olacağını belirlerse, kilitler için kendi alan ayırmak yerine bu işlemi işletim sistemine bırakır.

Çoğu durumda kilitleri dinamik olarak yönetme işin varsayılan değerde olduğu gibi SQL Server'a bırakılmalıdır. Eğer bu değer ayarlanmak istenirse (5000 - 2147483647 KB aralığı), SQL Server belleğin bu kısımlarını dinamik olarak yönetmez ve diğer alanlarda performans düşer. Eğer maksimum kilit sayısının aşıldığına dair bir hata mesajı alınırsa, şu seçenekler vardır: Tüm sorguların aşırı kilit oluşturup oluşturmadıkları kontrol edilmelidir. Eğer oluşturuyorlarsa uygulamadaki uyumluluk sorunu nedeniyle performans zarar görmektedir. Gereksiz yere çok fazla bellek ayırmaktansa sorguları düzeltmek daha iyi bir yoldur. Server'daki uygulama sayısı azaltılmalıdır. Server'a RAM takviyesi yapılmalıdır. Kilit sayısı daha yüksek bir değere çekilmelidir. Bu en istenmeyen çözümdür, çünkü bellek alanını, diğer SQL Server bileşenlerince kullanılmak yerine kilitlerin kullanımına ayırır.

### 3.4.8. Maksimum paralellik seviyesi (Max. degree of parallelism)

Bu seçenek eğer paralelliğin açık olup olmadığı ya da hangi CPU'larda açık olduğunun belirtilmesini sağlar. Paralellik bir sorguyu çalıştırmak için birden fazla CPU kullanmasıdır. Varsayılan olarak, paralellik açıktır ve affinity mask seçeneği kullanılmadıysa bir sorguyu çalıştırırken tüm CPU'lar kullanılır. Eğer tek CPU varsa "max degree of parallelism" değeri göz ardı edilir.

Varsayılan değer olan "0" paralelliğin açık olduğu anlamına gelir. Eğer bu değer 1 yapılırsa özellik kapatılır. Bu seçenek, paralellik için kaç CPU kullanılabileceğini tanımlamaya olanak verir. Örneğin 8 CPU varsa ve 4'ünün paralel çalışması isteniyorsa, bu değer 4 yapılmalıdır. Bu seçenek mümkün olsa da değişikliğin performans artışı getirmesi şüphelidir.

Paralellik açıksa, sorgu iyileştiricisi her bir sorgu için paralellik gerekliliğini araştırır, bu da biraz zaman alır. Birçok OLTP server'da, sorguların çok sık olması paralellik

kullanımını gerektirmez. Bu durumun örnekleri standart select, insert, update ve delete komutlarıdır. Bunun nedeni, sorgu iyileştiricisinin paralelliğin gerekli olup olmadığını her bir sorgu için araştırmakla vakit kaybetmek istememesidir. Eğer sorguların paralellik gerektirmeyeceği düşünülüyorsa, bu özellik performans düşünülerek kapatılmalıdır. Elbette, paralellik olması gereken bir özellik olup kullanılmak istenebilir. Eğer OLTP server'ı çok fazla bağlantılı ya da kompleks sorgu çalıştırıyorsa paralellik kullanılmalıdır.

Çoğu server OLTP ve OLAP sorgularını aynı anda çalıştırır ve paralellik açık olmalıdır. Denetimin bir parçası olarak, paralelliğin kapalı olduğu belirlenirse, nedeni araştırılmalıdır. Eğer sistem OLTP tabanlı ise, paralelliğin kapalı olması işe yarayabilir. Eğer OLTP ve OLAP karışıksa ya da çoğunluk OLAP ise paralellik açık olmalıdır.

### 3.4.9. Maksimum ve minimum sunucu belleği

En iyi performans için serverlar sadece SQL Server'a tahsis edilmeli ve bellek değerleri varsayılan değerlerde bırakılmalıdır. Varsayılan değerler, en iyi performans için bellek yönetimini SQL Server'a bırakır.

"Maximum server memory" ayarı varsayılan değeri olan 2147483647 MB'de bırakılırsa, SQL Server belleği dinamik olarak ayarlar ve ihtiyaç duyduğu kadarını kullanır. Eğer SQL Server'ın tüm belleği kullanması istenmiyorsa, bu değer manuel olarak değiştirilebilir. Girebilecek olan aralık 4 MB ile toplam RAM miktarı kadardır. Ancak işletim sistemine de ihtiyacı kadar bırakmak gereklidir.

Bazı durumlarda, SQL Server Server'daki diğer uygulamalarla aynı belleği kullanmak zorunda kalır. Örneğin diğer uygulamalardan biri SQL Server performansından çok daha önemliyse, SQL Server'ın kullanacağı RAM'i sınırlandırılabilir.

Bu değer manuel olarak değiştirildiğinde karşılaşılabilecek iki potansiyel performans sorunu vardır. İlki, SQL Server'a çok fazla RAM ayrıldığında işletim sistemine yeteri kadar bellek kalmayacağından sistemin genel performansı çok düşer. Ayrıca eğer Full-Text Search servisini kullanılıyorsa, yeterli miktarda bellek bu işlem için ayrılmalıdır. Çünkü bu servisin belleği SQL Server gibi dinamik olarak ayrılamaz ve düzgün çalışması için yeterli belleğe sahip olması gerekir.

"Min server memory" ayarı, varsayılan değeri olan 0'da ise, SQL Server belleği dinamik olarak ayırır. Bunun da anlamı, SQL Server'ın ihtiyacı kadar belleği değişen durumlara göre kullanmasıdır.

Eğer bu değeri varsayılandan farklı bir değer yaparsanız, bu değer otomatik olarak SQL Server tarafından kullanılmadığı zamanlarda bile işgal edilecektir.

Örneğin, 100 MB'lık bir minimum değer tanımlarsanız, SQL Server açılır açılmaz belleğin 100 MB'ını doldurur. Bu nedenle; bu değeri değiştirmek için hemen hemen hiçbir neden yoktur.

Eğer SQL Server'ınız sadece kendisine atanmış bir Server'daysa, bu değer değiştirilmemelidir. Eğer Server'da başka uygulamalar da çalışıyorsa, bu değeri değiştirmekte küçük yarar sağlanabilir. Ama bu değerin ne olduğu ve yarar sağlayıp sağlamayacağı konusu şüphelidir.

### 3.4.10. Maksimum metin değişikliği miktarı (Max. text repl. Size)

"Max text repl size" ayarı tek bir işlemde (insert, update, writetext veya updatetext) eklenebilecek maksimum metin ya da resim verisini belirlemenizi sağlar. Eğer replikasyon kullanılmıyorsa, bu ayarın varsayılan değeri en optimum değerdir.

Varsayılan değer 65536, minimum değer 0 ve maksimum değer 2147483647 KB'tır. Eğer text ya da image veri tipi verileriyle çok fazla replikasyon yapılıyorsa, bu değerin arttırılması düşünülebilir. Kesin değeri bulabilmek için değişik varyasyonları denenmelidir.

### 3.4.11. Maksimum işlem sayısı (Max worker threads)

"Max worker threads" ayarı işletim sisteminde sqlserver.exe işlemi tarafından çalıştırılan thread sayısını belirlemekte kullanılır. Varsayılan değer 255'tir.

Eğer 255'ten fazla kullanıcı bağlantısı varsa, birden fazla bağlantının aynı thread'i kullanıldığı durumlar için SQL Server thread pooling kullanır. Thread pooling SQL Server'ın kullandığı sistem kaynaklarını azaltsa da kullanıcı bağlantılarının çekişmesini artıracağından genel performansı düşürür.

SQL Server'ın kullandığı thread'lerin sayısını öğrenmek için Enterprise Manager kullanarak, bağlantı sayısını kontrol edilmelidir. "Max worker threads" sayısında belirtilen değere ulaşana kadar, her bir bağlantı için bir thread oluşturulur. Eğer 500 bağlantı varsa ve sadece 255 thread uygun durumdaysa, açık bağlantılar mevcut thread'leri paylaşır.

Server'da yeteri kadar RAM bulunduğunu kabul ederek, bu değeri sisteme bağlanabilecek maksimum kullanıcı sayısı artı beş olarak tanımlanmalıdır.

Önceden belirtildiği üzere bu ayarın varsayılan değeri 255'tir. Eğer 255'ten fazla bağlantı oluşmayacaksa bu değeri değiştirmeye gerek yoktur. Bunun nedeni, sisteme 50 bağlantı olsa bile 255 yerine 50 thread kullanılmasıdır.

Eğer 255'ten fazla kullanıcı sisteme bağlanıyorsa bu değer 255 ise, SQL Server thread pooling uygular. Sorun buradadır. Eğer yeteri kadar RAM varsa maksimum thread sayısını artırarak, SQL Server'ın ekstra kaynak kullanması sağlanabilir. Bu sayede genel performans da artar.

Eğer yeterli RAM yoksa, thread eklemek performansı düşürür. Bu durumda thread pooling daha iyi bir çözümdür. Çünkü thread pooling daha az kaynak kullanır. Diğer yandan thread pooling bağlantılar arasında kaynak çatışmasına neden olabilir. Örneğin aynı görevi aynı anda çalıştırmak isteyen ve aynı thread'i kullanan iki bağlantı çatışabilir.

Sonuç olarak yapılması gereken, eğere 255'ten az bağlantı varsa, değeri olduğu gibi bırakmak gerekir. Eğer 255'ten fazla ise ve ekstra RAM varsa bu değer, maksimum bağlantı sayısı artı 5 olarak değiştirilmeli. Eğer ekstra RAM yoksa, varsayılan değerde bırakılmalıdır.

### 3.4.12. Sorgu başına düşen minimum bellek miktarı (Min memory per query)

Bir sorgu çalıştığında SQL Server optimum bellek ayırmak ve en verimli ve en hızlı şekilde çalıştırmak için çalışır. Varsayılan değerde, sorgu başına minimum bellek 1024 KB'tır. Girilebilecek aralık 0- 2147483647 KB'tır.

Eğer bir sorgu verimli şekilde çalışmak için daha fazla belleğe ihtiyaç duyuyorsa SQL Server otomatik olarak daha fazla belleği ayırır. Bu nedenle bu ayarı değiştirmek tavsiye edilmez.

Bazı durumlarda eğer SQL Server ihtiyaç duyduğundan fazla RAM'e sahipse, bu değeri artırmak performansı artıracaktır. Örneğin 2048 KB ya da biraz daha fazla bir değer atanabilir. Fazladan bellek olduğu sürece bu değeri artırmak performansı artırır. Eğer fazladan bellek yoksa bu değeri artırmak performansı düşürür.

### 3.4.13. İç içe tetikleyiciler (Nested triggers)

Bu ayar performansı etkiler ama bilindik yoldan değil. Varsayılan değer olan 1 iç içe tetikleyicilerin çalışmasına (iç içe çalışan 32 tetikleyiciye kadar) izin verir. Eğer bu değer 0 yapılırsa, iç içe tetikleyicilere izin verilmez. Bu değer 0 yapılarak genel performans yükseltilebilir ancak uygulama esnekliği kaybedilebilir.

### 3.4.14. Ağdaki paketlerin büyüklüğü (Network packet size)

"Network packet size" SQL Server'ın ağ üzerinde client'larla bağlantı kurduğunda kullanacağı paket boyutunu belirler. Varsayılan değer 4096 byte'tır. Minimum değer 512 byte olup, maksimum değer kullanılan ağ protokolüne göre değişir.

Teoride bu değeri değiştirmek, eğer yeni değer paket boyutuyla uyuşuyorsa performansı artırır. Örneğin veri küçükse varsayılan değeri 4096 yerine 512 yapmak performansı artıracaktır. Ya da çok büyük datalarla işlem yapacaksanız, paket boyutunu artırmak, aynı datayı daha az seferde, daha büyük paketlerle göndermeyi sağladığından performansı artırır. Ancak gerçek hayatta performans artışı bu şekilde olmayacaktır. Çünkü ortalama paket boyutu diye bir kavram yoktur. Bazen datalar büyük, bazen de küçüktür. Bu nedenle varsayılan değeri değiştirmek pek kullanışlı değildir.

### 3.4.15. Açık nesneler (Open objects)

"Open objects" aynı anda açılabilen toplam nesne miktarını (tablo, görüntü tablosu, tetikleyici, depolanmış yordam vs. ) gösterir. Varsayılan değer olan 0, SQL Server'ın en iyi performans için bu sayıyı dinamik olarak belirlemesidir.

Nadiren server belleği çok doluysa, çok fazla objenin açık olduğu yolunda bir uyarı alınır. Buna en iyi çözüm server belleğini artırmak, ya da yönetilen veritabanlarını azaltmak gibi server üzerindeki yükü hafifletmektir.

Eğer hiçbiri yapılamıyorsa, "open objects" ayarı, uygun yüksek bir değere atanmalıdır. Burada dikkat edilmesi gereken iki nokta, uygun değerin ancak deneme yanılmayla

bulunabileceği ve açık objelere atanan hafızanın Server'ın genel performansını düşürebileceğidir. Bu değer değiştirildiğinde de sistem çalışır ama daha yavaş bir şekilde. Bu değerin değiştirilmesinden kaçınılmalıdır.

### 3.4.16. Sorgu maliyeti (Query governor cost limit)

"Query governor cost limit" bir sorgunun çalışabileceği maksimum sürenin belirlenmesini sağlar. Bazı kullanıcılar performansa zarar verecek kadar büyük ve uzun sorgular çalıştırıyorsa bu ayarı değiştirerek kullanıcıların örneğin 300 saniyeden uzun sorguları çalıştırmasını engellenmiş olur. Varsayılan değer olan 0, hiçbir limit olmadığını gösterir.

Bu ayar için girilecek olan değer sorgu iyileştiricisi tarafından tahmin edilen değerle bağlantılı bir değer olmalıdır. Eğer sorgunun çalışması için tahmin edilen süre bu değerden büyükse sorgu çalıştırılmayıp hata verir. Bu da server kaynaklarının boşa kullanımını önler.

Diğer tavsiyelerin aksine, eğer denetim esnasında bu alanda varsayılan değerden farklı bir sayıya rastlanırsa ve kullanıcılar şikâyet etmiyorsa bu iyi bir durumdur. Eğer 0 yazıyorsa, değer değiştirilip sonuçlar gözlemlenmelidir. Dikkat edilmesi gereken en önemli şey bu değerin çok küçük tutulmamasıdır. Örneğin 600 saniye ile başlayıp önemli bir sorunla karşılaşılmadığı sürece bu değer gittikçe düşürülebilir. Bu şekilde ideal süre bulunabilir.

### 3.4.17. Sorgu bekleme süresi (Query wait )

Eğer SQL Server çok yoğunsa ve daha fazla bellek istiyorsa, yeterli bellek bulana kadar daha fazla bellek isteyen sorguları yeterli bellek alana kadar sıraya koyacaktır. Bazı durumlarda yeterli bellek bulunmaz ve bu sorgular hata verir. Varsayılan değer olarak, sorgu iyileştiricisinin tahmin ettiği sürenin 25 katını aşarsa sorgu zaman aşımı hatası verir.

En iyi çözüm Server'a bellek takviyesi yapmak ya da yükünü azaltmaktır. Eğer bu yapılamıyorsa, bir seçenek, "query wait" konfigürasyon ayarı kullanılabilir. Varsayılan değer olan -1 yukarıda belirtilen zaman aşımı süresini kontrol eder, sonra hata verir. Eğer sorguların daha uzun bir süre sonrası hata vermesi isteniyorsa, bu değer daha büyük atanmalıdır.

Bu ayarı kullanmada çıkan zorluklardan biri, çok yoğun bir sorgu satırları kilitliyor olabilir ve bu da bununla ilgili bir soruna (deadlock gibi) yol açabilir. Bu nedenle bu ayar değiştirilmemelidir.

### 3.4.18. Minimum kurtarma aralığı ( Recovery interval-min )

Eğer SQL Server, çok fazla insert, update ve delete çalıştıran çok yoğun bir OLTP Server'ıysa , varsayılan "recovery interval" 0 olarak kalmalı, yani yaklaşık değeri SQL Server kendi belirlemelidir. Eğer performans, Performance Monitor üzerinden izleniyor ve işaretleme işlemleri sırasında açığa çıkan %100 diske yazma hareketliliği gözlemleniyorsa, bu değer 5 ya da 10 gibi bir değere yükseltilebilir. Bu süre, SQL

Server'ın yeniden başlatılmasından sonra çalıştıracağı bir kurtarma işleminin maksimum süresini dakika cinsinden verir. Varsayılan değer 0,kurtarma başına 1 dakika verir.

Bu ayarı kullanmanın bir başka nedeni, Server'ın bir OLAP ya da veri ambarı olarak kullanılma durumudur. Bu durumda, çoğu read-only veritabanı düşük kurtarma zamanıyla uyum sağlayamaz. Bu zamanı genişletilerek, SQL Server'ın işaretleme gerçekleştirme zamanını azaltır ve yükü hafifletilmiş olunur.

En uygun olan, bu zamanın olabildiğince küçük tutulmasıdır. Bunun nedeni SQL Server servisi her yeniden başlatıldığında otomatik kurtarma işlemi başlatır ve bu değer yüksekse daha fazla zaman alır.

Çok yoğun OLTP serverları için "recover interval" ayarının işe yarayıp yaramayacağı ancak uzun denemelerden sonra bulanabilir. Denemek çok önemlidir. Ama eğer server OLAP ya da veri ambarı ise bu ayarı artırmak kolay verilebilecek bir karardır.

### 3.4.19. Başlangıç işlemlerinin taranması (Scan for startup procs)

SQL Server eğer düzgün şekilde konfigüre edildiyse açılışta depolanmış yordamlar için tarama yapar. Eğer açılışta bir yordam çalıştırılması isteniyorsa bu özellik kullanışlıdır. Varsayılan değer 0, açılışta tarama yapılmayacağı anlamına gelir. Eğer açılışta çalışması istenen yordam yoksa bu şekilde bırakılmalıdır. Kullanılmayacak yordamlar için tarama yapmak gereksizdir. Eğer açılışta çalışması istenen yordam varsa bu değer 1 yapılmalıdır.

### 3.4.20. Bellek kümeleri limitlerinin belirlenmesi (Set working set size)

Bu seçenek SQL Server tarafından kullanılacak minimum ve maksimum bellek alanını belirlemek için kullanılır. Ayrıca sayfa takaslarını da engeller. Varsayılan değer "0" bu seçeneğin kullanılmadığını gösterir. Kullanmak için değeri “1” yapmalı, minimum ve maksimum değerleri aynı olarak girilmelidir. Bu değer çalışma düzeni boyutunu korumak için kullanılır.

Çoğu seçenek gibi bu da genelde gerekli değildir. Sadece Server’ın sadece SQL Server’a ayrılmış olup çok fazla RAM’e sahip olduğu durumlarda değerlendirilebilir. Bu durumda bile performans kazancı çok az olup işletim sistemine ayrılan bellek miktarını riske edebilir. Bu seçenek varsayılandan farklı belirlenmişse, minimum ve maksimum server bellek seçeneklerini de kontrol edilmelidir , aksi takdirde bu seçenek sağlıklı çalışmayacaktır.

### 3.4.21. Kullanıcı bağlantıları (User connections)

Varsayılan olarak SQL Server bağlanmak isteyen tüm kullanıcılara yer ayırır. Bu nedenle her yeni bağlanan kullanıcıda kullanıcı başına düşen bellek yeri azalır. "User connections" seçeneği varsayılan değer olan "0"daysa, kullanıcı bağlantıları dinamik olarak oluşturulur. Bu ideal seçimdir. Eğer bu değeri değiştirilirse SQL Server’a bağlanabilecek kullanıcı sayısı kesinleştirilmiş olur. Böylece kullanıcı başına ayrılan bellek alanı sabitlenmiş olur ve belirtilen sayıda kullanıcı bağlanmazsa bu alan sabit kalır. Bu değeri değiştirmeye bu nedenden dolayı gerek yoktur. Eğer "0"dan farklı bir şey yazıyorsa, “0” olarak değiştirilmelidir.

## 3.5. SQL Server Veri Tabanı Ayarları Kontrolü

Tablo 3.5. SQL Server ayarları kontrol listesi

| Veritabanı Konfigürasyonu Ayarları | Varsayılan Değerlerler | Şu andaki değerler |
|---|---|---|
| auto_close | Off | |
| auto_create_statistics | On | |
| auto_update_statistics | On | |
| auto_shrink | Off | |
| read_only | Off | |
| torn_page_detection | on in 2000<br>off in 7.0 | |
| compatibility level | 80 for 2000<br>70 for 7.0 | |
| Database auto grow | On | |
| transaction log auto grow | On | |

Performans denetiminin bir parçası olarak,server üzerindeki bütün veri tabanlarının üzerinde bazı temel veri tabanı ayarlarının incelenmesi gerekmektedir. Diğer veritabanı denetimi işlerine göre kıyaslandığında en basit denetim şimdi anlatılacak olan denetimdir. Bu denetimde kullanılacak olan temel tablo Tablo 3.5'tir.

İncelenmekte olan veri tabanı ayarları denetiminde 2 farklı ayar ele alınacaktır. Bunlar; veritabanı seçenekleri ayarları ve veritabanı konfigürasyon ayarları.

Hem veritabanı seçenekleri ayarları hem de veritabanı konfigürasyonu ayarları Enterprise Manager kullanılarak bir arayüz yapılabileceği gibi, query analyzer kullanılarak ALTER DATABASE komutu kullanılarak da yapılabilir.Bununla birlikte

veritabanı seçenekleri ayarları sp_dboption sistem depolanmış yordamıyla da ayarlanabilir. Burada bahsedilecek olan ilk bölüm veritabanı seçenekleri ayarlarına odaklanacaktır. İkinci bölüm ise veri tabanı konfigürasyon ayarlarına odaklanacaktır.

### 3.5.1. Veritabanı seçeneklerinin görüntülenmesi

Bu bölümde bahsedilecek olan ayarlar birçok veritabanı seçenekleri ayarları içinden performansı direkt olarak etkileyen ayarlardır. Bu ayarları görüntülemenin en iyi yolu Enterprise Manager'I kullanmaktır. Enterprise Manager kullanılarak sahip olduğu bütün veri tabanları görüntülenir. Ayarları gözlenmek isteyen veritabanı seçilerek üzerinde sağ tıklanır ve Özellikler alt menüsüne girilir. Özellikler alt menüsünden "seçenekler" menüsü seçilir. Bu ekrandan konuyla ilgili birçok veritabanı seçenekleri görülür. Ama bu arayüzde bütün veritabanı seçenekleri ayarlarının görünmediği unutulmamalıdır. Ancak performansa en çok etki eden ayarlar burada listelenmiştir.

#### 3.5.1.1. Otomatik kapanma (Auto_close)

Bu veritabanı seçeneği SQL Server 7.0 ve 2000'nin masaüstü versiyonları için oluşturulmuştur. Server'lar için bu ayar varsayılan olarak bulunmamaktadır. Bu yüzden açılmamalıdır. Bu ayar, veritabanına son bağlı olan kullanıcı bağlantısını kestiği zaman veritabanını kapatmaktadır. Ancak veritabanına yeni bir talep geldiği zaman veritabanı tekrar açılıp bu talebe cevap vermek durumundadır. Bu yeniden açılma zamanında iş yükü artar ve bu da zaman kaybına sebep olur.

Bu ayarın açık olmasının problem olması şu şekilde özetlenebilir. Eğer veritabanı çok sık talepler alan ve sıklıkla sorgulanan bir veritabanıysa bu açılıp kapanmalar veritabanı üzerinde yüksek performans düşüşlerine sebep olur.

Denetimin bu kısmında, SQL Server'ın masaüstü versiyonu kullanılmamasına rağmen bu ayar açık bulunursa bu ayarın neden açık olduğu araştırılmalıdır. Eğer sağlam bir gerekçe bulunmazsa bu ayar doğrudan kapatılabilir.

#### 3.5.1.2. Otomatik istatistik oluşturma (Auto_create_statistics)

Auto_create_statistics ayarı varsayılanda açıktır. Bunun açık olması durumunda "where" ifadesinin geçtiği sorgulardaki bütün kolonlar için otomatik olarak istatistik üretilir. Bir sorgu, sorgu iyileştiricisi tarafından ilk defa optimize edildiği zaman, bu kolonların daha önceden bir istatistiği olmadığı düşünülerekten bu istatistikler oluşturulur. Bu istatistikler sayesinde, sorgular çekilirken en optimum çalıştırma planları göz önünde bulundurularaktan cevap verilir. Bu sayede sorguların cevap süreleri kısalır.

Eğer bu ayar kapalı durumdaysa, bu istatistiklerin oluşturulması otomatiğe bağlanmamış olur. Ancak manuel olarak bu istatistikler istenirse oluşturulabilir. Bu ayarın kapalı olması durumunda sorgular için çıkarılan çalıştırma planları optimum çözümler ortaya koymaz…

Bu ayarın açık durmasının hiç bir olumsuz yönü yoktur. Kolon istatistiklerinin ilk defa çıkarılışında sorgunun çalışması çok kısa sayılabilecek bir seviyede uzayabilir ancak bu istatistikler bir defa çıkarsa, sorgunun bundan sonraki çalışmalarında kesinlikle çok daha verimli çalıştırma planları ortaya çıkacak ve sorguların cevap süreleri kısalacaktır.

Denetim sonucunda bu ayarın kapalı olduğu görülürse sebebi araştırılmalıdır. Eğer kuvvetli bir sebebi yoksa açılmalıdır.

### 3.5.1.3. İstatistiklerin otomatik olarak güncellenmesi (Auto_update_statistics)

Sorgu iyileştiricisinin akıllı sorgu optimizasyonları yapabilmesi için kolon ve indeks istatistiklerinin sürekli güncel olması gerekmektedir. Bunu garantiye almanın en iyi yolu bu veri tabanı seçeneği ayarını açık bırakmaktır(varsayılan değeri açıktır) Bu sayede iyileştiricinin kullandığı istatistiklerin geçerli olduğundan emin olunabilir.

Fakat bu ayar için, açık olması kesinlikle iyidir denilemez. Çok ağır iş yükü altındaki serverlarda, bu ayar büyük tablolardaki istatistikleri yanlış zamanlarda güncellemeye çalışabilir. Bu da zaten ağır olan server performansını iyice ağırlaştırabilir.

Denetim sonucunda bu özelliğin, Server'ın yoğun zamanlarında çalıştığı belirlenirse, bu özellik kapatılabilir. Tabloların kolonlarının ve indekslerinin istatistiklerinin güncellenmemesi eksikliği ise veritabanı daha az bir iş yükü altındayken "update statistics" komutu kullanılarak giderilebilir.

Ancak bu ayarın kapatılıp kapatılmaması için net bir şey söylemek mümkün değildir.Serverın çeşitli zaman aralıklarındaki meşguliyet durumu göz önünde bulundurularak bu özelliğin kapatılması durumunda, sorguların verdiği cevap süresinin uzaması durumuyla, açık olması durumunda yanlış zamanlarda çalışan istatistik güncellemesi olayının Server'a getirdiği ek yük durumu karşılaştırılıp çıkan sonuca göre bir karar vermek en doğru çözüm olacaktır.

#### 3.5.1.4. Otomatik küçülme (Auto Shrink)

Eski verileri silinen tabloların veritabanında ortaya çıkardığı boş alanlardan dolayı bazı veri tabanlarının periyodik olarak küçültülmesi gerekebilir. Ama sırf bunun için auto_shrink ayarı açılmaz. Çünkü bu ayar SQL Server kaynaklarını gereksiz yere boşa harcar.

Bu ayarın varsayılan değeri kapalı olmasıdır. Bunun anlamı veri tabanındaki boş alanları boşa çıkarmanın tek yolu bunun manuel olarak yapılmasıdır. Eğer bu özellik açılırsa SQL Server her yarım saatte bir boşaltılacak boş alan var mı diye kontrol eder. Başka yerlerde kullanılabilecek kaynakların bu işi için kullanılmasının yanında, otomatik küçültme işlemi hiç beklenmedik bir anda Server'da işlem yaparak server üzerinde darboğazlara sebebiyet de verebilir.

Eğer bu küçültme işlemi dönemsel olarak yapılmak isteniyorsa bunun "DBCC shrink database" veya "DBCC shrinkfile" komutlarının kullanılmasıyla veya SQL Server Agent'ın kullanılması veya veritabanı bakım planı oluşturulması daha iyi bir çözüm olacaktır. Denetim sonucunda eğer bu özellik açık bulunursa, sebepleri araştırılmalı. Eğer sağlam bir gerekçe yok ise kapatılmalıdır.

#### 3.5.1.5. Yalnızca okunabilirlik (Read_only)

Eğer veritabanı yalnızca veri çekimi için tasarlanmışsa bu özelliğin açık olması faydalı olacaktır. Örneğin yalnızca raporlama için kullanılan bir server bu durum için söz konusudur. Bu özelliği açık bırakmak bu gibi durumlarda performansı artırıcı etki yapar. Eğer veri tabanı üzerindeki verilerde nadir görülen değişiklikler oluyorsa bu

değişiklikler olurken bu özellik kapatılıp işlem bittiği zaman manuel olarak tekrar açılabilir.

#### 3.5.1.6. Yırtık sayfaların tespiti (Torn_page_detection)

SQL Server'ın veri sayfalarının 8K'lık olması, NT Server ya da Windows Server ailesi veri sayfalarının 512byte'lık olmasının uyumsuzluğu yüzünde ya da sabit disklerde yaşanan sorunlardan dolayı veri tabanında sorunlar ortaya çıkabilir.

Veri sayfası uyuşmazlığı sorunu şu şekilde oluşmaktadır. İşletim sistemi 8K'lık SQL veri sayfalarını kendi disklerine yazarken bu verileri 512 byte blokluk işletim sistemi veri sayfalarına dağıtır. Yani 1 bitin bir yarısı sabit diskin bir veri sayfasına yazılırken diğer kısmı da diğer yarım bit'lik kısma yazılacaktır. Eğer tam bu esnada bir elektrik gücü kaybı yaşanırsa SQL Server bu durumu algılayamaz ve 8Klık bütün verisinin yazıldığını düşünür. Herhangi bir geri sarma mekanizmasıyla bunu toparlayamaz. Bu durum da veri tabanı mimarilerinde yırtık sayfa olayı olarak bilinir.

Eğer sağlam kesintisiz güç kaynaklarıyla beslenmeyen serverlar mevcut ise ve bu tarz durumların yaşanmasından endişe edilirse bu özellik açılabilir. SQL Server bu tarz bir durum olduğunu fark ettiği zaman bunu onaramaz ama en azından en son yedekten geri dönüş yapılarak mevcut veriler kurtulabilir.

SQL Server 2000'de bu özelliğin varsayılan değeri açık olmasıdır. Bu durumda bu özelliğin her zaman açık olmasında bir sorun yok gibi görünebilir. Ama sorun şudur ki bu özelliğin açık durumda bırakılması SQL Server performansını etkiler. Normal çalışan bir SQL Server da bu hissedilmeyebilir ancak zaten performans olarak can çekişen bir

SQL Server'ımız var ise bazı güvenlik risklerini değişik yöntemlerle kapatarak bu özelliğin kapatılması performans olarak bir kazanç sağlayabilir.

### 3.5.2. Veritabanı konfigürasyon ayarlarının görüntülenmesi

Bu bölümde sadece 2 tane veri tabanı ayarından söz edilecektir. Bunları da görüntülemenin en iyi yolu Enterprise Manager'dır. Uyum seviyesini görüntülemek için Seçenekler bölümü, veritabanının otomatik büyüme ayarlarını görüntülemek için Veri Dosyaları bölümü ve işlem kayıtlarının otomatik büyümesinin ayarlarını görüntülemek için İşlem Kayıtı bölümü seçilebilir. Enterprise Manager kullanılarak sahip olduğu bütün veritabanları görüntülenir. Ayarları gözlenmek isteyen veritabanı seçilerek üzerinde sağ tıklanır ve Özellikler alt menüsüne girilir. Özellikler alt menüsünden, uyum seviyesini görüntülemek için Seçenekler bölümü, veritabanının otomatik büyüme ayarlarını görüntülemek için Veri Dosyaları bölümü ve işlem kayıtlarının otomatik büyümesinin ayarlarını görüntülemek için İşlem Kayıtı bölümü seçilebilir.

#### 3.5.2.1. Uyum seviyesi (Compatibility level )

SQL Server 2000, daha önceki versiyonlarıyla uyumlu çalışabilmesi için uyum modu moduna sahiptir. Bu şekilde örneğin 7.0 versiyonunda yazılmış depolanmış yordamlar, fonksiyonlar oluşturulmuş tablolar SQL Server 2000 üzerinde de çalışabilir. Eğer veritabanı için maksimum performans düşünülüyorsa bu ayara açık tutulmamalıdır.

SQL Server 7.0'ın uyum seviyesi 70, SQL Server 2000'in uyum seviyesi 80 ve SQL Server 2005'in uyum seviyesi 90dır.

### 3.5.2.2. Veritabanı ve işlem kaydı otomatik büyümesi

Veritabanı otomatik büyümesi ve işlem kaydı büyümesinin otomatik olup olmaması birlikte incelenmesi gereken iki ayardır. Çünkü birbiriyle çok yakından ilişkilidirler. Eğer SQL Server 2000 için bu ayarlar otomatik olarak büyümeye ayarlanmışsa ki varsayılan değeri budur, bu işlem her gerçekleşmeye kalktığında CPU'dan ve I/O süresinden kaynak almak ister. Bu işin en ideal olanı otomatik büyüme ne kadar az olursa bu için ayrılacak olan kaynaklarda o kadar az olur.

Bunu yapmanın bir yolu, veritabanı ilk oluşturulurken veritabanı boyutunun ve işlem kaydı boyutunun ulaşabileceği maksimum boyutları tahmin edip onlara en doğru boyutları vermektir. Teoride mümkün gibi görünse de pratikte böyle bir varsayımda bulunmak varsayımın ötesine geçmez.

Veritabanı ve işlem kaydı için varsayılan büyüme oranı %10dur. Bunun idealliği veritabanından veritabanına değişir. Eğer veritabanı çok fazla hit alıyor ve çok çabuk büyüyorsa bu oranı %20, %30 seviyelerine çekmek doğru olacaktır. Ancak unutulmaması gerek şey veri tabanı her büyümeye çalıştığında performansı aşağıya çeker. Bu yüzde de ne kadar çok verilirse veritabanının büyüme talebi o kadar az sıklıkla olacaktır. Eğer veritabanı 10 GB'dan daha büyükse yüzde vermek yerine veritabanının boyutunu sabitlemek daha iyi olacaktır. Çünkü büyük veri tabanının büyümesi de çok olacaktır. Örneğin 10 GB'lık bir veri tabanının %10 büyümesi 1 GB'lık bir operasyon iken bu daha küçük veri tabanlarında büyüme oranı daha küçük olacaktır. Eğer bu durum istenmeyen performans sıkışıklıklarına sebep oluyorsa bu durumda göre sabit büyümeler seçilebilir. Örneğin 100 MB 100Mb büyüsün diye bir seçenek seçilebilir. Ama yine bu seçimde veritabanının sorgu alma, ekleme ve güncelleme işlemleri alma oranına göre gözden geçirilmesi gereken bir durumdur.

## 3.6. SQL Server Veritabanı İndeks Performansı Kontrol Listesi

Tablo 3.6. SQL Server indeks performansı kontrol listesi

| Kontrol Listesi | Cevaplar |
|---|---|
| Son zamanlarda Indeks ayarlama sihirbazı (Indeks Tuning Wizard ) hiç çalıştırıldı mı? | |
| Her veritabanındaki bütün tabloların kümelenmiş-indeksleri var mı? | |
| Herhangi bir tablodaki kolonlardan birisi birden fazla indekslenmiş mi? | |
| Sorgularda kullanılmayan indeksler var mı? | |
| İndekslerden çok geniş olanlar var mı? | |
| Birleştirilen tabloların birleştirildikleri kolonlar üzerinde uygulanan indeksler doğru mu? | |
| İndeksler yararlı olmak için yeterince tekil mi? | |
| Kapsayan İndekslersin avantajları kullanılıyor mu? | |
| İndeksler ne kadar sıklıkla yeniden oluşturulma sürecine tabi tutuluyor? | |
| İndekslerin doluluk faktörleri ne? | |

SQL Server'da indeks denetimi zor bir iş olmasına karşın performansı yükseltmek açısından çok önemli bir iştir.

Yüksek miktarda indekslerin denetimini yapmak için 2 farklı yaklaşım mevcuttur. Birinci yaklaşıma göre; bütün iş küçük parçalara bölünüp indeksler gruplandırılır ve ilk önce toplam server performansına en çok etkide bulunacak indekslerin denetimine odaklanılır. Örneğin bu yaklaşıma göre, SQL Server'daki en meşgul veritabanının en büyük veriyi tutan tablosundan denetime başlanır küçük tablolara doğru ilerlenir. Bu şekilde başlangıçtaki çaba SQL Serverın performansına en büyük pozitif etkiyi yapacak veri tabanı ve tablolar üzerinde gösterilir.

Diğer yaklaşım ise hata yönetimi yaklaşımıdır. Buna göre normal bir zamanda server üzerindeki indeksleri değerlendirmenin çok bir faydası yoktur. Bu indekslerle

ilgilenilmesi gereken ve çaba gösterilmesi gereken zamanlar Server'ın performans sıkıntısı gösterdiği zamanlardır. Bu zamanlarda bu indekslerin davranışlarını gözlemlemek, yapılan değişikliklerin nasıl etki yaptıklarını görmek için yeterince fırsat vardır. Ve yine bu yaklaşım türünde de indeks denetimine en büyük sıkıntıya sebebiyet verebilecek olan geniş indekslerden başlanır.

Yaklaşım her ne olursa olsun indeks denetimi işlemi öncesinde sistematik bir planla yaklaşmak gerekir. Çünkü indeksler üzerinde yapılacak değişiklikler performansa çok ciddi etkiler yapar.

### 3.6.1. İndeks ayarlama sihirbazının çalıştırılma sıklığı

MS SQL Server 7.0 ile gelen ve SQL Server 2000'de daha gelişmiş bir şekilde ortaya çıkan İndeks Ayarlama Sihirbazı sayesinde kullanılmayan indeksler, kullanılması durumunda sorguları hızlandıracak olan indeksler belirlenebilir. İndeks Görüntüleme de veritabanında kullanılan sorguları kullanarak veritabanının nasıl kullanılması gerektiğine dair önerilerde bulunan güzel bir araçtır. Analiz etmek için kullanması gereken sorgular, SQL Server Profiler aracından gelir.

Acil bir veritabanı performans analizinde yapılabilecek ilk adım server aktivitelerini profiler gibi bir araçla izlemek ve sonuçlarını indeks ayarlama sihirbazına vermektir. Bu sayede kullanılmayan ve silinebilme ihtimali olan indeksler ile yeni oluşturulması gereken indeks önerilerine hızlı bir şekilde ulaşılabilir.

Aşağıda SQL Server'ın indeks denetimi yapılırken kullanılabilecek olan indeks ayarlama sihirbazının kullanımına dair bazı ipuçları verilmiştir.

İndeks ayarlama sihirbazının kullanacağı verileri yakalayan ve ona sunan profiler aracı veritabanı üzerinde, o veritabanının bir günlük gerçek yükünü sunabileceği saatlerde ve sürede çalıştırılmalıdır. Yani veri tabanının normal bir zamandaki aktivitelerinin fotoğrafını çekebilmelidir.

Veriler profiler tarafından izlenip kaydedildikten sonra indeks ayarlama sihirbazı istenilen herhangi bir zamanda çalıştırılabilir. Ama tabi ki doğru olanı Server'ın çok yoğun olmadığı zamanlarda çalıştırılmasıdır. Bunun sebebi de, indeks ayarlama sihirbazı tarafından gerçekleştirilen analiz sırasında server üzerinde yük binmesi oluşturacaktır. Buna bir alternatif olarak şu da yapılabilir; sihirbazın ana Server'a bağlantısı olmak koşuluyla analiz işi başka bir test Server'ında yaptırılabilir. Bu sayede ana server üzerinde yük bir oranda azalır.

Analiz süresini uzatacağı kesin olmasına karşın analizin daha doğru veriler üretebilmesi açısından sihirbaza ilk başta bazı seçenekler sunulmalıdır. Bunlar; Mevcut "bütün indeksleri muhafaza et" (Keep all existing indekses) seçeneği seçilmemelidir. "Örnekleme için kullanılacak olan ağır sorguların sayısını limitle" (Limit the number of workload queries to sample) seçeneği seçilmelidir. "indeks başına kolon sayısını maksimize et" (maximize columns per indeks) ayarı en fazla 16 olarak seçilmelidir. Ayarlama için de bütün tablolar seçilip belirtilmelidir. Bu ayarları belirtmek sihirbazın analiz süresini uzatacak olmasına karşın tam bir analiz yapmasına neden olur. Aynı zamanda profiler tarafından alınıp indeks ayarlama sihirbazına verilen kayıtların büyüklüğü, sihirbazın çalışacağı server makinesinin donanım konfigürasyonu bu süreye olumlu ya da olumsuz anlamda etki edecektir.

Analiz bittikten sonra, sihirbaz herhangi bir öneride bulunamayabilir, bir veya birden fazla indeksin kaldırılmasını önerebilir, yeni bazı indeksler eklenmesini önerebilir veya hepsini bir arada önerebilir. Bu öneriler gerçekleştirilmeden önce dikkatlice

değerlendirilmelidir. Örneğin sihirbazın kaldırılmasını istediği bir indeks aslında gerçekten tek bir iş için önemli bir indeks olabilir. Bu durumda sihirbaz neden bu indeksin silinmesini önerir? Bunun sebebi sihirbaz profiler'dan gelen bütün kayıtları analiz etmez bunlardan bir örnekleme yapar. Hepsini analiz etse dahi bu indeksin önemli rol oynadığı sorgu o sırada profiler'ın kayıtlarına düşmeyebilir.

Yeni bir indeks eklenmesi önerisi geldiği zamanda da bu indeksin diğer indekslerle uyumlu çalışıp çalışmayacağını düşündükten sonra eklemek gerekmektedir. Örneğin tavsiye edilen bir indeks bir sorgu için gerçekten performans arttırırken saatte binlerce defa gelen bir INSERT komutunun işlemesini yavaşlatabilir. Bunu sihirbaz bilemez ve gelen sihirbaz önerilerini değerlendirmek ve devam eden günlük işlerdeki önceliklere göre neyin yavaş neyin hızlı olmasının performansı daha arttıracağını düşünüp ona göre indeks ekleyip eklememek yine bu kararı alacak olan kişiye bağlıdır.

Son olarak da sihirbaz yaptığı analiz sonucunda hiç bir öneride de bulunmayabilir. Bu da her şeyin mükemmel olduğu anlamına gelmez. Yine sorun yaratabilecek sorgular profiler'ın kayıtlarına düşmemiş olabilir. Bunun için aslında profiler'ın kayıt toplama işlemini günün haftanın değişik saatlerinde yaparak mümkün olan bütün örnekleme kombinasyonlarına indeks ayarlama sihirbazına sunmak gerekmektedir.

Bir def analiz yapılıp da gerekli değişiklikler yapıldıktan sonra işlem bitmemiştir. Profiler düzenli aralıklarla veri toplaması yapıp sihirbaza bu verileri vermeli ve yeni analizler ürettirip yapılan değişikliklerin gerçekten işe yarayıp yaramadığı gözlemlenmelidir. Ayrıca veri tabanı sürekli sabit kalan bir yapıda değildir. Veri eksilmesi veya artmasının yapılan değişikliklerden nasıl etkilendiği de mutlaka gözlemlenmelidir.

### 3.6.2. Tabloların kümelenmiş indekslerinin incelenmesi

Bilinen genel bir kural olarak bütün tabloların en az bir tane kümelenmiş indeksi bulunmalıdır. Genellikle kümelenmiş indeksler özdeşlik kolonu gibi sürekli artan kolonlar üzerinde olmalıdırlar ve tekil olmalıdırlar. Birçok durumda kümelenmiş indeksler için en uygun kolon birincil anahtar kolonlarıdır.

SQL Server 6.5 da performans ayarlama sihirbazı kullanılırken verilen bir öneri vardı. Kümelenmiş indekslerinizi sürekli artan kolonlara koyarsanız performans sıkıntıları yaşayabilirsiniz çünkü diskiniz üzerinden "hotspot" diye tabir edilen sıcak noktalar oluşturabilirsiniz deniyordu. 6,5 versiyonu içinde bu önerme doğruydu.

Ancak SQL Server 7.0, 2000 ve 2005'de bu sıcak noktalar bir sorun olmaktan çıkmıştır. 1 saniye önce hot spotların negatif yönde etkilemesini beklediğimiz ardışık 1000 işlem gerçekleşmiş olabilir. Ancak aslında bu tarz durumlarda sayfa bölünmelerine sebep olduğu için hot spotlar yararlıdır

Bunun sebebine gelince; eğer bir tablo üzerinde ardışık olarak artan özelliğe sahip birincil anahtar özelliğinde bir kümelenmiş anahtar var ise bu tablo üzerine gelen INSERT komutlar fiziksel disk üzerinde ardışık eklemeler yapacaktır. Bu yüzdende sayfa bölünmeleri olmayacaktır. Bu yüzden de hem yazarken hem de bu verileri ararken disk üzerinde fazla bir işlem yapılmayacaktır. Bu da performans artışına sebep olacaktır.

Eğer bir yığındaki(kümelenmiş indeksi olmayan tablolar) birçok satırdan söz edilecek olunursa verilerin eklenirken herhangi bir ardışık sırada eklenmez. Bazı kolonların artışı ardışık olsa da ardışık sırada eklenmez. Bu da SQL Server'a o veriler için bir sorgu geldiği zaman daha çok okuma işlemi yapmasına sebebiyet verir. Öte yandan bu

tablonun ardışık olarak artan kolonuna bir tane kümelenmiş indeks koyulursa artık bu verileri yazmak ve çekmek için daha az I/O disk işlemime ihtiyaç duyulacaktır.

Toparlanacak olursa ardışık olarak özellikle çok insert, update ve delete komutları alan tablolarda ardışık olarak artan kolonlara kümelenmiş indeksler koyulması performansı toplamda olumlu yönde etkileyecektir. Ama eğer tablo üzerinde veri modifikasyonundan çok seçim işlemleri yapılıyorsa bu öneri daha az faydalı olabilir.

Sonuç itibariyle indeks denetimin bir parçası olarak her tabloda indeks olup olmadığı kontrol edilmeli. Eğer herhangi birisinde indeks yoksa uygun olan bir kolona en azından bir tane kümelenmiş indeks koyulması en kolay çözüm olarak düşünülebilir. Teoride hiç bir indeksi olmayan bir tabloya kümelenmiş bir indeks koymanın hiç bir olumsuz etkisi olmadığı unutulmamalıdır.

### 3.6.3. Birden fazla indeksleme yapılan tabloların bulunması

Bu öneri hakkında fazla düşünülecek bir şey yoktur. Yapılan araştırma sonucunda ortaya çıkan sonuca göre yapılması gereken şey kesindir. Eğer veritabanı üzerinde çalışan birden fazla kişi var ise ve aynı kolon üzerine farklı isimlerde aynı özelliklere sahip indeksler koyulmuşsa SQL Server bunun ne amaçla yapıldığını bilemez ve bir uyarı vermeden bunu kabul eder. Bu gerek manuel olarak yapılan çalışmalarla gerekse de indeks ayarlama sihirbazının yaptığı analizin sonucunda bulunursa direkt olarak bu indekslerden fazla olanları silmek gerekmektedir. Bunu yapmak disk üzerinde alan açmakla kalmaz veriye erişimi de modifikasyonunu da kolaylaştırır.

Bu soruna sebep olan en alışılageldik yanılgı birincil anahtar özelliğindeki kolonların zaten otomatik olarak kümelenmiş indekse sahip olmalarının gözden kaçırılmasıdır.

Bunun sonucunda da başka bir isimle aynı özelliğe sahip bir indeks oluşturmaya çalışmaktır.

### 3.6.4. Kullanılmayan indekslerin bulunması

Bu sorunun cevabına göre yapılacak olanlar da çok açıktır. Ancak bir önceki maddedeki gibi bu da çok sık karşılaşılan bir sorundur. Tabloların indekslerini bulup onlara bakarak bunların sorgularda kullanılıp kullanılmadığı anlaşılmaz. Bunun anlaşılması için indeks ayarlama sihirbazı kullanılabilir. Eğer bu tarz indeksler bulunmuşsa bunları kaldırmak kesin doğrudur. Aynen tekrarlayan indekslerde olduğu gibi sorgularda kullanılmayan indeksleri kaldırmak da disk alanından tasarruf sağlayacağı gibi verilere erişimi ve düzeltilmesi işlemlerini de kolaylaştıracaktır.

### 3.6.5. Geniş indekslerin bulunması

Bir indeksin geniş olması onun fiziksel olarak da daha fazla yer tutması anlamına geldiği gibi, veriye erişimde ve düzenlenmesinde SQL Serverın daha fazla performans göstermesine sebep verir. Bu yüzden geniş kolonlara indeks koymaktan kaçınılması gerekmektedir. Dar indekslerin üzerinde yapılan işlemler daha hızlıdır.

Birden fazla kolonun birlikte birincil indeks oluşturduğu Bileşik İndekslerden mümkün olduğunca kaçınmak gerekmektedir. Eğer bir veritabanında yoğun olarak bileşik indeks kullanılıyorsa bu veri tabanının tasarımında bir sorun olduğu kolaylıkla söylenebilir.

### 3.6.6. Birleştirilen tabloların birleştirildikleri kolonlar üzerindeki indekslerin incelenmesi

Birleştirilen tabloların birleştikleri kolonları üzerinde indeksler bulunması tartışılmayan bir gerçek ve hatta uygulanması gereken bir kuraldır. Ancak en uygun birleştirme performansı için kullanılan indekslerin denetlenmesi kolay bir işlem değildir.

Birleştirilmek istenen ikin tablo arasında birincil anahtar ve yabancı anahtar ilişkisi kurulurken yabancı anahtar olarak belirlenen kolon üzerinde yabancı anahtar indeksi oluşturulmaması yapılan en genel hatadır. Birincil anahtar oluşturulduğu zaman o kolon üzerinde kümelenmiş indeksin otomatik oluşturulduğunun bilinmesi birleştirilen tablolardaki ikincil anahtar olarak belirlenen kolon üzerinde de otomatikman indeks oluşturulduğu yanılgısına düşülmesine sebep olur. Eğer bu tarz bir işlem yapılmışsa yabancı anahtar indeksi manuel olarak oluşturulmalıdır.Bunun sık sık unutulmasından dolayı, denetim sırasında bütün birincil ve ikincil anahtarların listesi çıkarılıp üzerlerinde uygun indekslerin olup olmadığı kontrol edilebilir.

Bütün bunların ötesinde, indeks ayarlama sihirbazı da unutulmuş olan birleştirme indekslerinin bulunmasında işe yarar.

### 3.6.7. İndekslerin tekilliğinin incelenmesi

Bir tabloda birden fazla indeks var diye SQL Server Sorgu Analizcisi üzerindeki bütün sorguları kullanacak anlamına gelmez. Kullanılmadan önce sorgu analizcisi bunların sorgu için faydalı olmadığına karar vermek durumundadır. Eğer bir tablo içindeki veri o tablo içinde en az %95 oranında tekil değilse sorgu analizcisi o kolondaki o kümelenmemiş indeksi kullanmaz. Bu yüzden tekilliği en az %95 olmayan kolonlara

kümelenmemiş indeksler koyulması anlamsızdır. Örneğin bit veri tipine sahip yani değeri yalnızca 1 ya da 0 olabilecek olan bir kolona kümelenmemiş indeks koymak mantıklı değildir. Bu denetim kuralından da görüldüğü üzere tablolar üzerindeki indekslere bakmanın yanı sıra verileri de incelemek faydalı olacaktır.

### 3.6.8. Kapsayan indekslerin incelenmesi

Kapsayan indeks bileşik indeksin bir çeşididir ve SELECT, JOIN ve WHERE ifadelerinde kullanılan bütün kolonları kapsayan indekstir. Bu yüzden aranılan veriyi bu indeks zaten içerir ve SQL Server veriyi aramak durumunda değildir. Bu da fiziksel ve mantıksal diskler üzerindeki iş yoğunluğu azaltıp performansı arttırır. Kapsamayan bileşik indeksler performansı düşürürken, kapsayan bileşik indeksler de çok yararlı olabilir ve birçok durumda sorgunun performansını arttırır.

Bu işin zor kısmı ise kapsayan indekslerin belirlenmesidir. İndeks ayarlama sihirbazı buna da bir çözüm önerisi getirmesine karşın her zamanki riski yine söz konusudur ve kapsayan indeks koyabilme fırsatı olan bazı kolonları kaçırabilir. Ancak bu aracı kullanmamanın alternatifi her bir sorguyu bireysel olarak inceleyip buna karar vermektir. Büyük veri tabanlarında da bu iş gerçekten yapılması neredeyse olanaksız olan bir şeydir.Bu durumda denetimin bu kısmında amaç, yeni kapsayan indeksler bulmak değil, bunların varlığından haberdar olup, gerektiği yerlerde performans açısından olumlu yönde kullanılabilecek şekilde kullanmaktır.

### 3.6.9. İndekslerin yeniden oluşturulma frekanslarının ele alınması

Zamanla parçalanan indeksler SQL Server'ın bunlara erişimini zorlaştırır ve performans üzerinde çok ciddi olumsuz etkiler oluşturur. Bunun çözümü ise belirli aralıklarla indekslerin birleştirilme işleminde geçmesidir.

Denetimin bu aşamasındaki amaç; veritabanındaki indekslerin birleştirilme işleminden geçip geçmediğinin ve eğer geçiyorsa bunun işleme frekansının ne olduğunun bulunmasıdır. Bu işlemin günlük, haftalık ya da aylık olarak zamanlaması yapılabilir. Bunun kararını verirken de veritabanının ne kadar değiştiği ne kadar modifikasyon komutları aldığı göz önünde bulundurulması gereken kriterdir. Eğer bir veritabanı çok fazla modifiye oluyorsa normalden daha fazla birleştirme işlemine tabi tutulması gerekmektedir. Ancak eğer veritabanı çok büyükse birleştirme işlemi çok zaman alır. Ve de Server'ın kaynaklarını yoğun bir şekilde kullanır. Bundan da canlı ortamda bulunan bir veri tabanında kullanıcılar olumsuz yönde etkilenirler. Denetimin bu aşamasında birleştirme işleminin mevcut sistemde ne kadar sıklıkla yapıldığı sorgulanmalı ve bunun optimum zamanının ne olduğu belirlenmelidir.

### 3.6.10. İndekslerin doluluk faktörlerinin incelenmesi

İndekslerin yeniden oluşturulması süreci doluluk faktörü konsepti ile birbirine çok yakındırlar. Yeni bir indeks oluşturulduğu zaman veya mevcut bir indeks yeniden oluşturulduğu zaman, indeksler tarafından doldurulacak olan veri sayfalarının ne kadarının dolu olduğunu belirten doluluk faktörü özelliği belirtilir. Bir doluluk faktörünün 100 olması demek her bir indeks sayfasının tamamının dolu olduğu anmalına gelirken 50 olması yarı yarıya dolu bir şekilde indeks sayfalarının oluşturulacağını ifade eder.

Eğer %100 doluluk faktörüyle bir kümelenmiş bir indeks oluşturulursa bu kolona eklenen her bir veri ayrı bir veri sayfasına yazılacaktır. Çünkü doluluk faktörü ile bu sayfa dolu olsun ya da olmasın tamamı doludur diye bir işaret konulmuştur. İşte bu yüzden de her bir veri eklenişinde sayfa bölünmeleri ortaya çıkacaktır. Meydana gelen birçok sayfa bölünmesi SQL Serverın performansını yavaşlatır.

Bir örnek vermek gerekirse; Varsayılan doluluk faktörüyle bir tablo üzerinde yeni bir indeks oluşturulduğu varsayılsın. İndeks oluşturulurken, SQL Server bu indeksi komşu fiziksel sayfalara yazmak isteyecektir. Veri dizi halinde okumak isteyeceği için optimum bir I/O erişimi sağlanmış olacaktır. Ancak tablo üzerinde veriler değiştikçe, silindikçe, yeni veriler eklendikçe sayfa bölünmeleri oluşacaktır ve bu sebepten dolayı SQL Server yeni sayfaları farklı fiziksel alanlara yazmak durumunda kalacaktır. Bu yüzden bu verilere erişim yapılırken rasgele bir I/O erişimi kullanılmak zorunda kalacaktır. Bu da hem verinin istendiği zaman erişimini yavaşlatacaktır.

Bu durumda ideal doluluk faktöründen bahsetmek mümkün mü? Bunu belirlemek tablolara yapılan yazma ve okuma işlemlerinin oranına bağlıdır. Düşük güncellemeye sahip tablolar:100 defa okuma 1 defa yazma (100/1) oranına sahip tablolarda %100 doluluk faktörü belirlenebilir. Yüksek güncellemeye sahip tablolar: Yazma işlemi sayısı okuma işlemi sayısını geçen tablolardır. %50 ile %70 arasında doluluk faktörü verilebilir. Ortalama olarak birbirine eşit oranlarda yazma ve okuma oranına sahip tablolar: %80-%90 arasında doluluk faktörü belirlenebilir. Bu kurallar genel olarak yapılan incelemelerden sonra ortaya çıkmıştır. Ancak optimum değerler bulmak yine bir SQL Server'dan diğerine göre değişiklik gösterir. Hiç bir zaman düşük doluluk faktörü belirlemek her zaman iyidir diye düşünülemez. Düşük doluluk faktörleriyle sayfa bölünmelerinde azalma olmasına rağmen bu defa da SQL Serverın okuması gereken veri sayfası sayısı artacaktır.

Düşük doluluk oranıyla azalacak olan sadece I/O aktivitelerinin performansı değil aynı zamanda tampon bellek de bu düşük orandan etkilenecektir. Veri sayfaları diskten tampona hareket ettiği zaman bir anda birçok veri sayfası tampona geçecektir. Bu da tamponda aynı zamanda bulunması gereken ve önemli olabilecek verilerin tampona alınması için gerekli olacak boş alanı azaltacaktır.

Eğer herhangi bir doluluk faktörü belirlenmezse varsayılan doluluk faktörü değeri sıfırdır. Doluluk faktörünün sıfır olması demek %100 doluluk oranı demektir ( indekslerin yaprak sayfaları %100 dolu olmasına rağmen ara katman veri sayfalarında hala biraz yer kalmıştır)

Sonuç olarak da denetim sürecinin bir parçası olarak yeni bir indeks oluşturulurken veya indeksler yeniden oluşturulurken doluluk faktörü oranı belirlenmesi gereken bir kriterdir. Teoride sadece okuma yapılan veri tabanları haricinde 0 doluk faktörü(bir diğer deyişle %100 doluluk faktörü) hiç bir zaman uygun değildir. Bunun yerine uygun bir miktar boş alan bırakacak bir doluluk faktörü belirlenmelidir.

## 3.7. SQL Server Uygulaması ve Transact-SQL

Tablo 3.7.1. Transact SQL kontrol listesi

| **Sorular** | **Cevabınız** |
|---|---|
| Transact-SQL kodu gerekenden fazla kod döndürür mü? | |
| İmleçler ihtiyaç duyulmadıklarında da kullanılmaktalar mı? | |
| UNION ve UNION SELECT uygun şekilde kullanılmaktadırlar mı? | |

Tablo 3.7.1. Transact SQL kontrol listesi (Devam)

| | |
|---|---|
| SELECT DISTINCT uygun şekilde kullanılmaktadır mı? | |
| Geçici tablolar gerekmedikleri halde kullanılmaktalar mı? | |
| İpuçları sorgularda öncelikli olarak kullanılmaktalar mı? | |
| Görüntü tablolarının kullanımı gereklimidir? | |
| Depolanmış yordamlar mümkün olduğu her durumda kullanılıyor mu? | |
| Depolanmış yordamlar içinde, SET NOCOUNT ON kullanılmakta mı? | |
| Herhangi bir Depolanmış yordam sp_ ile başlamakta mı? | |
| Tüm depolanmış yordamlar sahibi DBO mu ve [veritabanısahibi].[nesne_ismi] formunda mı? | |
| Kısıtlar veya tetikleyiciler gösterimle ilgili bütünlük için kullanılıyor mu? | |
| İşlemler mümkün olduğunca kısa tutulmaktalar mı? | |

Tablo 3.7.2. Uygulama kontrol listesi

| | |
|---|---|
| Uygulama, SQL Server ile iletişimde depolanmış yordam mı, Transact-SQL kodu mu yoksa ADO gibi bir nesne modeli mi kullanmaktadır? | |
| Uygulama SQL Server ile iletişimde hangi metodu kullanmaktadır: DB-LIB, DAO, RDO, ADO, .NET? | |
| Uygulama SQL Server ile iletişimde ODBC mi OLE DB mi kullanmaktadır? | |
| Uygulama, bağlantı havuzlamanın avantajından yararlanmakta mıdır? | |
| Uygulama bağlantıları uygun şekilde açıyor, tekrar kullanıyor ve kapatıyor mu? | |
| SQL Server'a gönderilen Transact-SQL kodu, SQL Server için en optimum kod mu yoksa jenerik SQL kodu mudur? | |
| Uygulama, SQL Server'dan ihtiyacından çok veri döndürmekte midir? | |
| Uygulama, kullanıcı veri düzenlerken işlemleri açık bırakmakta mıdır? | |

SQL Server optimizasyonunu olumsuz olarak etkileyen tüm kısıtların içinde, SQL Server verisine erişmek için kullanılan ve Transact-SQL kodu içeren uygulama kodu, optimizasyona negatif etki yapabilecek olan en büyük potansiyele sahiptir. Maalesef, bu birçok veritabanı yöneticisinin direkt olarak kontrol edemediği bir alandır ve bu yüzden, bu alan SQL Server tabanlı uygulamalardaki optimizasyon ayarlamalarında sık sık ihmal edilir.

Tabii ki, 3. parti yazılım kullanılıyorsa, kodla ilgili fazla bir şey yapılamayacağı için performans optimizasyonu için kod anlamında yapılabilecek pek bir şey kalmamaktadır. Fakat uygulamalar kontrol altında geliştirilmişse optimizasyon için yapılabilecek bir şeyler var demektir.

Optimizasyon öğeleri ve aşağıda belirtilen detayları gözden geçirildiği zaman bu bölümleri belirlemenin veya onarmanın küçük bir iş olmadığı kolayca anlaşılacaktır. Bu yüzden, uygulamanın yazıldıktan sonra onarımlar yapılması yerine, uygulamanın bu performans ipuçlarını akılda tutularak geliştirilmesi çok daha iyidir.

### 3.7.1. Transact-SQL kontrol listesi

Transact-SQL kontrol listesi 13 alt bölüm altında incelenecektir. Bu bölümlerden her bir tanesinde, performans yönetimi için t-sql komutlarında nelere dikkat edilmesi gerekliliği anlatılacaktır.

#### 3.7.1.1. Transact-SQL kodunun döndürdüğü veri miktarının incelenmesi

SQL Server'dan ne kadar az veri döndürülürse, SQL Server o kadar az kaynak kullanır ve bu da SQL Server'ın tüm optimizasyonunun artmasına yardımcı olur. Bu aşikâr olarak gelebilir fakat gereksiz veri döndürülmesi bir optimizasyon sorunudur.

Aşağıda kodlayıcılar tarafından SQL Server'dan veri döndürürken gerekenden daha fazla veri döndürülmesine yol açan en çok yapılan yanlışlar bulunmaktadır:

WHERE cümleciği eksikliği. Genelde az rastlanan tablodan tüm verileri döndürmek işlemi istenmedikçe, WHERE cümleciğinin kullanılması, dönen satır sayısının azaltılmasını sağlayacaktır.

Yukarıdaki tavsiyeye ilave olarak, bir WHERE cümleciğinin olabildiğince seçici olması gerekmektedir. Örnek olarak, bir tarihin sadece belli günlerine ait kayıtları döndürülmesi gerektiğinde, ay veya yıla ait kayıtların döndürülmesi gereksizdir. . WHERE cümleciğini sadece döndürülmesi istenen satırları döndürecek şekilde tasarlamak gereklidir.

Select cümleciğinde hiçbir zaman "select *" ifadesi şeklince kullanılmamalıdır. Select cümlesi görüntüleme tabloları üzerinde yapılmaktansa gerçek tablolar üzerinde yapılmalıdır.

Bunlardan haberdar olunmadığı durumlarda, gereksiz veri döndürmenin oluşturduğu bazı optimizasyon konuları şöyledir; bazen çok fazla veri döndürmek sorgu iyileştiricisini indeks taraması yerine tablo taramasına yönlendirir. Veriyi okumak için ilave I/O gerekir. SQL Server tarafından başka amaçlarla daha iyi kullanılabilecek olan önbellek alanı boşa kullanılır. Gereksiz ağ trafiği oluşur. Kullanıcıda, başka kullanımlar için gerekebilecek bellekte ilave veri depolanmış olur.

### 3.7.1.2. İmleçlerin kullanımı

Herhangi tip imleçler SQL Server'ın optimizasyonunu yavaşlatır. Bazı durumlarda göz ardı edilemez olsalar dahi, birçok durumda göz ardı edilebilirler. Bu yüzden uygulama Transact-SQL imleçlerini kullanmaktaysa, kodun bunlardan kaçınılacak şekilde tekrar yazılabilirliğine bakılması uygun olacaktır.

Satır – satır işlemler uygulamak istenildiğinde, imleç kullanmak yerine şu seçeneklerden bir veya fazlasının kullanılması göz önünde bulundurulabilir. Geçici tablolar kullanılması , while döngülerinin kullanılması , türetilmiş tablolar kullanılması , ilişkili alt-sorgular kullanılması, case ifadelerinin kullanılması, çoklu sorgular kullanılması.

### 3.7.1.3. Union ve union select kullanımının incelenmesi

Genel olarak union ve union select'in nasıl çalıştığını tam olarak bilinmez ve bu da SQL Server'ın birçok kaynağının gereksiz yere harcanmasına neden olur. UNION kullanıldığı zaman, sonuç kümesinde bir select distinct'in eşitliğini gösterir. Bir başka deyişle, bir UNION iki benzer kayıt kümesini birleştirir ve daha sonra olası aynı kayıtları araştırır ve onları teke indirir. Eğer amaç buysa, UNION kullanmak doğrudur.

Fakat UNION ile birleştirilmek istenen iki kayıt kümesinde aynı kayıtlar yoksa UNION kullanmak kaynakların boşa kullanılmasına yol açar, çünkü olmayan aynı kayıtları araştıracaktır. Bu yüzden birleştirilmek istenen kayıt kümelerinde aynı kayıtların olmadığı biliniyorsa, UNION yerine UNION ALL kullanması gerekir. UNION ALL kayıt kümelerini birleştirir fakat SQL Server kaynaklarını azaltan, server optimizasyonunu azaltan işlem olan aynı kayıtları araştırmayı gerçekleştirmez.

### 3.7.1.4. Select distinct'in kullanımı

Optimizasyon açısından, "distinct" cümleciği, sadece özel işlevi olan bir kayıt kümesinden aynı kayıtları ortadan kaldırmak istendiğinde kullanılmalıdır. Çünkü "distinct" cümleciğinin kullanılması sonuç kümesinin sıralanması ve aynı olanların ortadan kaldırılmasını gerektirir ve bu işlem de SQL Server'ın kaynaklarının önemli bir miktarını kullanılmasını gerektirir. Tabii ki bunu yapmaya ihtiyaç varsa yapılmalıdır. Fakat eğer "select" ifadesinin hiçbir zaman aynı kayıt döndürmeyeceği biliniyorsa, distinct cümleciği kullanmak sadece gereksiz bir şekilde SQL Server'ın kaynaklarının kullanılması demektir.

### 3.7.1.5. Geçici tablolar kullanımının incelenmesi

Geçici tabloların birçok kolay kullanımları olmasına rağmen, imlecin ortadan kaldırılması gibi, ek yükleri de bulunmaktadır ve bu ek yük ortadan kaldırılabilirse, SQL Server daha hızlı çalışacaktır. Örnek olarak, ek yükün azaltılmasını ve optimizasyonun artmasını sağlayacak olan, geçici tabloların ortadan kaldırılabilmesi için çok çeşitli yollar bulunmaktadır. Geçici tabloları ortadan kaldırmak için mevcut yollardan bazıları şöyledir; kodun tekrar yazılması, böylece istenilen işlem, standart bir sorgu veya depolanmış yordam kullanarak tamamlanabilir. Türetilmiş bir tablo kullanılarak ortadan kaldırılabilir. İlişkilendirilmiş alt-sorgular kullanılarak , kalıcı tablolar kullanılarak, geçici bir tabloyu taklit etmek için bir "union" ifadesi kullanılarak ortadan kaldırılabilir.

### 2.7.1.6. İpuçlarının sorgularda kullanılma durumunun incelenmesi

Genel olarak, SQL Server Query Optimizer sorguları en iyileştirme açısından iyi işler. Fakat bazı nadiren görülen durumlarda, Query Optimizer iş üzerinde başarısız olur ve

sorgudan en iyi sonucu elde etmek ve Query Optimizer'ı önemsiz kılmak için bir sorgu ipucu gerekir.

İpuçları bazı durumlarda yararlı olabilirlerse de, bazen de tehlikeli olabilirler. Bu yüzden, ipuçlarının kullanımı büyük bir dikkatle yapılmalıdır.

En önemli konulardan bir tanesi ipuçlarının büyük ölçüde kullanımını sağlayan bazı kodları miras almaktır, özellikle SQL Server 6.5 veya SQL Server 7.0 için yazılmış olan ve SQL Server 2000 altında çalışan kodu. Birçok durumda, SQL Server'ın önceki versiyonlarında gereken ipuçları yeni versiyonlarda uygulanamazlar ve kullanımları optimizasyona yardımcı olmaz, aksine zarar verir.

Bir başka durumda, muhtemelen ipuçları, ilk olarak uygulama ilk kez çalıştırıldığında kullanışlı olarak algılanır, fakat zaman geçtikçe ve saklanan verinin "yapısı" değiştikçe, kullanışlı ipuçları bir kez "yeni" veriye daha fazla uygulanamadıkça optimizasyona potansiyel olarak tehlike arz ederler.

Her iki durumda da, kullanılan sorgu ipuçlarını periyodik olarak yeniden değerlendirmek iyi bir yaklaşımdır. Mevcut ipuçlarının o kadar da kullanışlı olmadıkları görülebilir ve ayrıca optimizasyonu da kötü yönde etkiler. Bunu bulmanın tek yolu da onları sorgu analizcisinde test etmek, tam olarak ne olduğunu görmek ve daha sonra da sonuca göre onları kullanmaya devam edip etmeme kararını vermektir.

### 3.7.1.7. Görüntü tablolarının kullanımının incelenmesi

Görüntü tabloları en iyi olarak, tembel geliştiricilerin yöntemi olarak sık kullanılan sorguları saklamak için değil, güvenlikle ilgili konularda kullanılmaktadırlar. Örnek olarak, bir kullanıcıya SQL Server verisine anlık sorgu erişimine izin vermek için, bu kullanıcı (veya grup) için bir görüntü tablosu oluşturulması düşünülebilir, daha sonra da bu kullanıcıya tablolara değil de view'e erişim için izin verilir. Öte yandan, uygulamada görüntü tablolarından veri seçmenin iyi bir gerekçesi yoktur, bunun yerine istene veri tablolardan seçilmelidir. Görüntü tabloları gereksiz yere ek yük oluşturur ve birçok durumda, gerektiğinden daha fazla veri döndürülmesine yol açar.

Örnek olarak, iki birleştirilmiş tablodan 10 sütun döndüren bir görüntü tablosu olduğu ve görüntü tablosundan bir SELECT ifadesiyle 7 sütunun alınmak istendiği düşünülsün. İlk olarak görüntü tablosunda sorgu çalışır ve veri döndürür, daha sonra sorgudan dönen veri üzerinde sorgu çalışır. Bu şekilde görüntü tablosundan dönen 10 sütun yerine 7 sütuna ihtiyaç olduğundan dolayı, gerekenden daha fazla veri döndürülmüş olur ve SQL Server kaynaklarını boşa kullanılmış olur. Uygulamalarda takip edilmesi gereken verilerin mümkün olduğunca görüntü tablolarından değil, tablolardan çekilmesi gerekliliğidir.

### 3.7.1.8. Depolanmış yordamların kullanım yerlerinin incelenmesi

Depolanmış yordamların kullanıcılarına sağladıkları birçok yarar vardır. Bunlar; uygulama optimizasyonunu artıran ağ trafiğini ve gecikme süresini azaltır. Örnek olarak, ağa 500 satırlık t-sql cümleciği göndermek yerine, çok daha hızlı ve daha az kaynak kullanan depolanmış yordam kullanılmalıdır. depolanmış yordam çalışma planı SQL Serverın belleğinde tutularak tekrar kullanılabilir, böylece Server'ın ek yükünü

azaltır. Kullanıcı çalışma istekleri ise daha etkindir. Örnek olarak, bir uygulama büyük ikili değeri bir kopya veri sütununa depolanmış yordam kullanmadan INSERT etmek isterse, ikili değeri karakter satırına (boyutunu ikiye katlar) çevirmelidir ve SQL Server'a göndermelidir. SQL Server onu alınca, tekrar karakter değer formatından ikili formatına çevirmelidir. Bu da birçok ek yük demektir. Bir depolanmış yordam bu konuyu ortadan kaldırır, çünkü parametre değerleri uygulamadan SQL Servera kadar tüm yol boyunca ikili formatında kalır, ek yük azalır ve optimizasyonu artırır. Depolanmış yordamlar ilerlemiş kodun tekrar kullanımına yardımcı olur. Bu, direkt olarak uygulamanın optimizasyonunu artırmasa da, geliştiricilerin üretkenliğini gereken kod miktarını ve hata ayıklama zamanını azaltarak artırabilir. Depolanmış yordamlar mantığı özetlerler. Depolanmış yordam'ın kodu, kullanıcıları etkilemeden değiştirilebilir (parametrelerin değiştirilmediği ve hiçbir sonuç kümesi sütununun kaldırılmadığı varsayılarak). Bu da geliştiricinin zamanından tasarruf eder. Depolanmış yordamlar verilere daha fazla güvenlik sağlar. Eğer depolanmış yordamlar yalnız kullanılıyorsa, tablolardan select, insert, update ve delete haklarını direkt olarak kaldırabilir ve geliştiriciler, veri erişimi için depolanmış yordamları kullanma yöntemine yönlendirilebilir.

Genel bir kural olarak, tüm sql cümleleri kodu depolanmış yordamlardan çağrılabilmelidir.

### 3.7.1.9. Depolanmış yordamlar içinde, set nocount on komutunun kullanımının incelenmesi

Varsayılan olarak, bir depolanmış yordam her çalıştırıldığında, depolanmış yordamdan etkilenen satır sayısını içeren bir mesaj Server'dan kullanıcıya gönderilir. Bu bilgi kullanıcı için nadiren yararlıdır. Bu varsayılan özelliği kapatarak, server ve kullanıcı arasındaki ağ trafiği azaltılabilir ve böylece Server'ın ve uygulamanın tüm

optimizasyonunun da artırılması sağlanmış olur. Bu özelliği depolanmış yordam seviyesinde kapatmak için, "set nocount on" komutu kullanılmalıdır.

### 3.7.1.10. Depolanmış yordamların isimlerinin incelenmesi

Eğer ana veritabanı dışında bir veritabanında çalıştırmak üzere bir depolanmış yordam oluşturuluyorsa, isminde ön ek olan "sp_" ön eki kullanılmalıdır. Bu özel ön ek, sistem depolanmış yordamları için ayrılmıştır. Her ne kadar bu ön eki kullanmak bir kullanıcı tanımlı depolanmış yordamının çalışmasını engellemezse bile çalışmasını bir miktar yavaşlatır.

Bunun nedeni varsayılan olarak, "sp_" ön ekiyle başlayan ve SQL Server tarafından çalıştırılan herhangi bir depolanmış yordam, ilk olarak ana veritabanından çözülmeye çalışılır. Orada olmadığı için, bu depolanmış yordamın yerine bakılarak boşa zaman harcanmış olur.

Eğer SQL Server depolanmış yordamı ana veritabanında bulamazsa, daha sonra depolanmış yordam ismini, nesnenin sahibi "dbo"'umuş gibi düşünür. Depolanmış yordamın mevcut veritabanında olduğunu varsayarak çalıştırır. Bu gereksiz gecikmeyi engellemek için, hiçbir depolanmış yordamı ön ek olan "sp_" ile adlandırmamak lazım.

### 3.7.1.11. Depolanmış yordamların nesne sahipliğinin incelenmesi

En iyi optimizasyon için, aynı depolanmış yordam tarafından çağrılan tüm nesnelerin sahibi aynı olmalıdır ve dbo tercih edilir. Eğer değillerse, nesne isimleri aynı fakat sahipleri farklıysa, SQL Server isim kararlılığı yapmalıdır. Bu olduğunda, SQL Server

"bellek planından" bir depolanmış yordam kullanamaz, onun yerine, depolanmış yordamı tekrar derlemelidir ve bu da optimizasyonu engeller.

Uygulamalardan bir depolanmış yordam çağırılırken, onu tam ismiyle çağırmak da önemlidir. Exec myProcedure yerine Exec dbo.myProcedure gibi. Bunun optimizasyonla ilgili birtakım nedenleri vardır. İlk olarak, tamamen isimleri kullanmak, hangi depolanmış yordamın çalıştırılacağı konusunda olası herhangi bir karışıklığı ortadan kaldırır, hataları ve diğer olası problemleri önler. Fakat daha önemlisi, bunu yapmak SQL Server'ın depolanmış yordamların çalışma planına direkt olarak erişimine imkân sağlar ve buna bağlı olarak, depolanmış yordamın hızını arttırır. Evet, optimizasyon artırımı çok küçüktür, fakat eğer server her saatte on binlerce veya daha fazla depolanmış yordam çalıştırıyorsa bu tip ufak zaman kazanımları arttırabilir.

#### 3.7.1.12. Kısıtlar veya tetikleyicilerin bütünlüğünün incelenmesi

Veritabanını gereksiz bütünlük özellikleri eklememelidir. Örneğin, gösterimle ilgili bütünlüğü sağlamak için birincil anahtar ve yabancı anahtar kısıtları kullanılıyorsa, aynı işleve sahip olan tetikleyiciyi de ekleyerek gereksiz ek yük yaratılmamalıdır. Aynısı, kısıtları ve varsayılanları veya kısıtları ve gereksiz iş sağlayan kuralları kullanmak için de geçerlidir.

#### 3.7.1.13. İşlemlerin çalışma sürelerinin incelenmesi

Tüm t-sql cümlecikleri mümkün olabildiğince kısa tutulmalıdır. Bu kilitlerin sayısını azaltılmasına yardımcı olur ve SQL Serverın optimizasyonunu da artırmayı sağlar. Eğer uygulanabilirse, uzun cümleleri küçük gruplar halinde cümlelere bölmek yöntemi uygulanabilir.

## 3.8. SQL Server Görev Optimizasyon Denetimi

Tablo 3.8. SQL Server görevler kontrol listesi

| SQL Server Görev Doğrulama Listesi | Cevaplar |
|---|---|
| Çalışan ama gereksiz olan bir görev var mı? | |
| Görevler üretim hareketsizliği süresince çalışmaları için zamanlanmışlar mıdır? | |
| Aynı Server'daki görevler çakışmaktadırlar mı? | |
| SQL Server görevi olmayan ve çakışan herhangi bir görev var mı? | |
| T-SQL çalıştıran görevler Optimize Edilmişler midir? | |
| Görevlerin ne kadar çalıştıkları kontrol edildi mi? | |
| Mevcut görevlerin alternatifleri var mıdır? | |

Her SQL Server sanal olarak bir veya daha fazla görevi günlük ve birçok görevi de haftalık olarak çalıştırabilir.

Herhangi bir uygulamanın SQL Server'ın optimizasyonunu olumsuz etkileyebileceği gibi, aynı durum görevler için de geçerlidir. Zayıf tasarlanmış kodları çalıştıran veya kötü zamanlarda çalıştırılan görevler SQL Server'a önemli bir engel koyabilirler. Bu yüzden, SQL Server'ın görevlerini optimizasyon denetimi olarak dâhil etmek önemlidir.

SQL Server optimizasyon denetiminin bu bölümünde, olası görevle ilgili optimizasyon konularının nasıl belirlenmesi ve düzeltilmesi üzerine odaklanılmıştır.

### 3.8.1. Çalışan görevlerin önem derecelerinin belirlenmesi

Görevler ilk defa çalıştırılıp zamanlandıktan sonra bir süreliğine takip edilirler. Eğer doğru çalışıyorlarsa bir süre sonra unutulurlar. Örneğin, rapor oluşturmak için geceleri çalışmak üzere bir görev oluşturulabilir ve birçok tablodan başka bir tabloya veri taşır. Fakat bu rapor daha fazla kullanılmayacaksa, bu görevin de daha fazla çalıştırılmasına gerek yoktur ve ek yükü azaltmak için kaldırılmalıdır. Problem ise, görev ve rapor arasında direkt olarak bir bağlantı yoktur, böylece rapor daha fazla kullanılmıyorsa görevin kaldırılması da kolayca unutulabilir.

Denetiminizin parçası olarak, her bir Server'da çalışan her bir görevi tekrar gözden geçirmek ve görevin gerçekten yararlı olup olmadığını belirlemek gerekir.

### 3.8.2. Görevlerin zamanlamalarının gözden geçirilmesi

SQL Server'da çalışan her bir görevi tekrar gözden geçirirken, ne zaman çalıştıklarını da yakından incelenmeli. Bir görevin özel bir zamanda çalışmasının gerekmediği varsayılarak, görevleri zamanlarken SQL Server'ın daha az yoğun olduğu zamanlar, duruma göre hafta sonları veya geceleri seçilmelidir.Yani genel olarak görevlerin çalışmaları için üretim ortamındaki SQL Serverın en hareketsiz olduğu zamanlar belirlenip o zamanlarda çalıştırılmalıdır.

### 3.8.3. Çakışan görevin olup olmadığının kontrol edilmesi

SQL Server'daki birçok aktiviteler için, görevlerin bir kerede tamamlanması yerine, olabildiğince zamana yayılması idealdir. Örneğin, SQL Server'da 10 veritabanı varsa ve

her biri için yedekleme görevleri oluşturulmuşsa, hepsini aynı zamanda çalıştırmak yerine, her birinin birer zamanda çalışmak üzere zamanlanması daha iyidir.

Enterprise Manager'dan bir görevin ne kadar çalışacağı görülebilse de, görevleri çakışmayacak şekilde biri diğerinden sonra çalışacak şekilde (her bir göreve tamamlanması için yeterli zamanı vererek) el ile zamanlamanın kolay bir yolu yoktur. Bu yapılabilir, fakat birçok görevli serverlar için hepsini görüntülemek için bir elektronik çizelge gerekebilir. Bir seçenek olarak, tüm görevlerin görsel olarak görülebilmesini ve yönetmesini sağlayan ve kritik görevlerin örtüşmeyeceğini garanti eden bir 3.parti program kullanılabilir. Örnek vermek gerekirse kullanımı kolay ve güvenilir olan SQL Sentry programı kullanılabilir.

### 3.8.4. SQL Server görevleri dışındaki zamanlanmış görevlerin incelenmesi

Server'daki SQL Server görevlerinin yanı sıra, SQL Server olmayan görevler de olabilir. Bunlara örnek olarak birleştirme içeren veya SQL Server zamanlayıcısını kullanmayan disk yedekleme görevleri verilebilir. Bunlar SQL Server zamanlayıcısını kullanmadıkları için, kolayca unutulabilir ve bu görevlerin bazılarının aynı zamanda çalışmasını durdurabilir. SQL Server görevleri için, bu görevler SQL Server görevlerinin çalışmadığı zamanlarda çalışmak üzere zamanlanabilirse ideal olur .

### 3.8.5. T-SQL çalıştıran görevlerin optimizasyonu

Uygulamalarda ve script'lerdeki kod gibi, bir görevin parçası olarak çalışan T-SQL'de optimum olmalıdır. Görev kodunun olabildiğince etkin çalışabilmesi için ilgili indeksleri de eklenmelidir.

Böylece T-SQL kodu içeren her görev için, çalışma planını görmek, olası problemlere bakabilmek için sorgu analizcisinde çalıştırmak gerekir . Ayrıca indeks sihirbazı ile optimizasyonu artırmak için olası yararlı indekslere bakılabilir.

### 3.8.6. Görevlerin çalışma sürelerinin belirlenmesi

Her bir ayrı görevin ne kadar çalıştığını gözlemlemek için Enterprise Manager kullanılabilir. Fakat farkında olunması gereken husus bir görevin farklı zamanlardaki çalışma sürelerinin farklı olabileceğidir. Örneğin, özel bir görevin çalışması normalde 2 dakika sürer, fakat haftada bir kez Pazar günleri aynı görevin çalışmasının 15 dakika sürebilir. Bir görevin çalıştırılmasında, zaman miktarındaki önemli değişiklikler bu görevle ve SQL Server'da çalışan diğer bir işlemle ilgili çakışma olduğunun bir işaretidir.

### 3.8.7. Görevlerin alternatiflerinin belirlenmesi

Sadece çalışan bir görevin olması, görevi başarıyla tamamlamanın en iyi yolu olduğu anlamına gelmez. Her bir görev ele alınıp aynı işi yapmanın daha iyi bir yolu olup olmadığı belirlenmelidir. Örneğin, T-SQL kodu yazmak, muhtemelen mevcut DTS paketini kullanmaktan daha etkin olacaktır. Bir diğer örnek, SQL Server tarafından çalıştırılan bir görev, onun yerine bir başka zamanlayıcı program tarafından yapılabilir. Hatırlanması gereken önemli nokta, mevcut görevlerin tek çözüm olmadıkları ve server ek yükünü azaltan daha iyi çözümlerin mevcut olabileceğidir [7]

## 3.9. Profiler Kullanımı

Tablo 3.9. SQL Server sorgu performansı kontrol listesi

| SQL Server İş Kontrol Listesi | Cevaplar |
|---|---|
| Çok uzun süre çalışan bütün sorgular tanımlandı mı? | |
| Sorguların önceliği belirlendi mi? | |
| Önceliği belirlenen bu sorguların çalıştırma planları gözden geçirildi mi? | |

### 3.9.1. Uzun süre çalışan sorguların belirlenmesi

SQL Server performans denetiminin bu adımında depolanmış yordamlar da dâhil olmak üzere yavaş çalışan bütün sorguların tanımlanması ve SQL Server kaynaklarının adilce kullanılması sağlanmaya çalışılmalıdır.

Cevabı verilmesi gereken ilk soru uzun sorgunun süresinin ne olduğudur. Kimi uygulamalar için 3 saniyeden geç cevap veren bir sorgu uzunken kimi uygulamalar için 15 dakikada cevap veren bir sorgu uzundur. Bu göreceli yapıdan dolayı bu adımda ilk yapılması gereken şey; beklenen sorgu süresinin ne olduğuna karar vermektir.

Bu denetim süresince kullanılması uygun olacak araçlardan birisi MS SQL Server 2000 üzerinde gelen Profiler aracıdır. Profiler tarafından veri toplama işlemi başlatılmadan önce şu maddeler göz önünde bulundurulmalıdır; gözlemin yapıldığı server ile profiler'ın beri topladığın server aynı olmamalıdır. Bu Server'ın performansını negatif bir şekilde etkiler. Profiler çalıştırılırken ihtiyaç olandan başka veriler toplanmamalıdır. Ne kadar fazla veri toplanması hedeflenirse kullanılacak olan kaynak da o kadar

artacaktır. Veri toplama işlemi için sadece ihtiyaç olan veri kolonları ve olaylar seçilmelidir. Veri örneklemesinin sağlıklı olması ve gerçeği yansıtması için profiler'ın çalışma zamanı, tipik bir iş sürecinin gerçekleştiği saatler arasında olmalıdır. Bunun için günün farklı saatlerinde de veri toplanabilir. Ayın farklı zamanlarında da veri toplanabilir. Örneğin ay sonu faaliyet raporunun yavaş çalışması beklendiği zaman ay sonlarıdır. Ayın ilk haftalarında bu raporu çalıştırıp bunun verileri profiler'dan toplamak gerçeği yansıtmayacaktır [8] .

Profiler kullanılırken 2 seçenek vardır. Birincisi grafik kullanıcı arabiriminin kullanılması,ikincisi ise profiler için hazır olan sistem depolanmış yordamlarının kullanılmasıdır.

### 3.9.2. Toplanacak olan verilerin belirlenmesi

Profiler hangi olayların ve hangi verilerin toplanması seçiminin yapılmasına olanak sağlar. Bir diğer deyişe bütün verilerin gelmesi istenilen filtreye göre engellenebilir. Sorguların performansı için toplanılması önerilen olaylar şunlardır;Stored Procedures--RPC:Completed, TSQL--SQL:BatchCompleted .Sorguların performansı için toplanılması önerilen veriler ise şunlardır;

-Süre (Duration. Verilerin gruplanması için ihtiyaç duyulan süre)

- Olay sınıfı (Event Class)

- Veritabanı ID'si (databaseid. Eğer server üzerinde birden fazla veritabanı varsa)

- Metin Verisi (textdata)

- İşlemci (CPU)

- Yazma işlemi (writes)

- Okuma işlemi (reads)

- Başlama süresi

- Bitiş süresi

- Uygulamanın Adı

- NT kullanıcı adı

- Login adı

- PID

Kullanılacak filtreler ise şunlardır;

- Süre > 5000 millisaniye (Yani çalışma süresi 5 saniyeyi geçecek olan sorgular)

- Sistem olayları verileri toplanılmamalıdır.

- Bir seferde server üzerindeki bütün veritabanlarına ait veriler toplanmamalıdır. Tek tek her veritabanının verisi toplanmalıdır. Böylelikle Verilerin süzülmesi işlemi daha kolay olur.

### 3.9.3. Verinin toplanması

Kullanılan filtrelere göre, profiler'ın çalıştığı süreye göre, bu süre içerisinde server üzerindeki aktivitelerin yoğunluğuna göre elde edilen verilerin miktarı değişecektir.

Profiler verileri topladıktan sonra isteğe bağlı olarak en uzun süre çalışan sorgudan daha aza göre bir sıralama yapacaktır.

### 3.9. 4. Verilerin analiz edilmesi

Profiler'a verilmiş olan filtreler sayesinde aslında uzun sorgular ortaya çıkmış oldu. Dolayısıyla ara yüzden 5 saniyeyi seçilirse artık sadece çalışması 5 saniyenin üzerinde olan sorgular listelenecektir. Buradan elde edilen sonuçlara göre sorguların önem sırası ve kullanılma frekansına göre bir planlama yapılıp bu sorguların düzeltilmesine gidilebilir. Uzun sorguların önceliklendirilmesinde en uzun sorgulara öncelik verilmesi en doğru yaklaşım olacaktır demek yanlış bir yaklaşım olabilir.O en uzun sorgu yılda bir defa çalıştırılan bir sorgu olabilir. Ancak ondan daha kısa ama yine de 5 saniyeden fazla süren bir sorgu günde kullanıcılar tarafından sürekli çalıştırılan ve server performansını yoran bir sorgu olabilir. Bir diğer deyişle, ne kadar sürmesinin yanında ne sıklıkla çalıştırıldığı verisi bir arada analiz edilirse önceliklendirme işi daha doğru bir şekilde yapılabilir [9] .

### 3.9.5. Sorguların çalıştırma planlarına göre analiz edilmesi

Profiler tarafından, belirtilen filtrelere ve özelliklere göre yakalanan sorguların sorgu analizcisi aracına götürüp orada çalıştırıp, çalıştırılma planlarının incelenmesi gereklidir. Eğer yakalanan sorgu direkt t-sql sorgusu ise kopyalama yapıştırma metoduyla taşınabilir ancak bir depolanmış yordamın içinde geçen cümleyse bu iş biraz daha zorlaşır. Çünkü profiler bu cümlenin yordam içindeki hangi cümle olduğunu belirtmeden direkt olarak yordamın adını ve aldığı parametrelerin listesini verir. Bu tarz bir durumda yordamın içeriğini sorgu analizcisine kopyalayıp parametreleri verip çalıştırmak ve cümleleri tek tek incelemekten başka yapılacak bir şey yoktur.

### 3.9.6. SQL Server sorgu analizcisi çalıştırma planı analizleri

Sorguların çalıştırma planlarını görüntülemenin çeşitleri yöntemleri vardır. Bunlar; sorgu analizcisi içerindeki "Çalıştırma Planını Göster" seçeneğini seçerek olabilir. Eğer sorgu çalıştırmak istenmeden sadece hesaplanan çalıştırma planı görüntülenmek isteniyorsa; üst menüdeki Sorgu bölümün altındaki "Hesaplanan Çalıştırma Planını Göster" seçeneği seçilmedir. Birinci seçenek ile arasındaki fark, birinci seçenek çalışırken çalıştırma planına server üzerinde o andaki diğer aktivitelerin de etkisi olur. Ancak sadece hesaplanan plan gösterilirken dış ortamdan izole edilmiş bir yapı düşünülür. Ancak genel anlamda birçok durumda aynı çalıştırma planı üretilir. SQL Server Profiler aracı çalıştırılırken seçilebilecek olaylardan bir tanesi de MISC: Çalıştırma Planı'dır.Metin formundaki bilgi sorgu iyileştiricisinin kullandığı çalıştırma planını gösterir. Sorgu analizcisi aracı içindeyken "set showplan_text on" komutu çalıştırılaraktan burada çalıştırılmış olan herhangi bir sorgu sonrasında sorgu planı metin formatında verilir.Eğer sorgularda geçici tablolar kullanılıyorsa, "set statistics profile on" komutu,sorgu çalıştırılmadan önce çalıştırılmalıdır

Bu seçeneklerden Çalıştırma Planını Göster" seçeneği en kullanışlı olanıdır. Hem grafiksel olarak hem de metinsel olarak bir sonuç vermektedir.

Eğer aşağıdaki maddeler çalıştırma planında görüntüleniyorsa bunları bir uyarı olarak algılamak gereklidir. Bu maddelerin hepsi performans açısından idealin altındaki değerleri işaret etmektedir.

-İndeks veya tablo taramaları: Daha iyi bir indeks veya ek bir indeks ile bu sorun giderilebilir.

-Yer İmi Aramaları: Mevcut kümelenmiş indeks değiştirilerekten veya kapsayan bir indeks kullanılaraktan veya seçime giren kolon sayısını azaltaraktan üstesinden gelinebilir.

-Filtre**:** Where ifadesindeki kullanılmış olan bir fonksiyon var ise bunlar kaldırılarak, kullanılmış olan görüntü tablosu var ise bunlardan vazgeçerek veya ek indeks kullanarak kurtulunabilir.

-Sıralama : Sıralama çok gerekli değilse sorgu içerisindeki Order By ifadesi kaldırılabilir. İllaki sıralama gerekliyse bu sıralamayı sağlamak için bir indeks kullanılabilir. Ya da sıralama son kullanıcıya sunulan arayüzde kodlamayla yapılabilir. Çalıştırma planlarının sonuçlarını okuyup analiz etmek çok basit bir işlem değildir. Bu işlemler yapılırken şunlar göz önünde tutulmalıdır**:**

Çok karmaşık sorgu planları bir kaç parçaya bölünebilir. Her bir parça sonuca ulaşmak için sorgunun sırasıyla çalıştırılması gereken adımlarını içermektedir. Her bir çalıştırma planı adımı genellikle daha küçük alt birimlere bölünürler. Ancak bu alt birimler ana birimlerin soldan sağa okunması gibi değil,sağdan sola sıralamasıyla okunur. Yani o alt adımın başladığı nokta en sağdaki görünen birimidir. Her bir adım ve alt adım bir ok ile birbirine bağlıdır. Bu ok da sorguların çalıştırılma sırasını gösterir. Adım veya alt adımların üzerine gelinirse, bir açılır-kapanır pencere açılaraktan o adımla ilgili daha çok detayı içeren bir bilgi sunar. Adımları birbirine bağlayan oklar üzerine gelince de bu adım sonrasında kaç tane kaydın bir önceki adımdan etkilenerek yeni adıma geçtiğine dair bilgi verilir. Bir adımı diğerine bağlayan oklar farklı kalınlıklarda olabilir. Bu okların kalınlığı her bir adım arasındaki satır sayısı ve satır büyüklüğünün hesaplanmasının maliyetinin birbirine göre karşılaştırılmasına göre değişir. Örneğin daha kalın oklar ekranın solunda değil sağında görülmelidir. Eğer sol taraftaki adımlarda bu oklar görünüyorsa bu planın optimal plandan uzak olduğu anlaşılabilir. Bir çalıştırma

planında her bir adımın bir maliyet yüzdesi vardır. Yani o adımın yapılması toplam işlemin yüzde kaçını oluşturduğu bu şekilde belirtilir. Çalıştırma planları analiz edilirken yüksek yüzdeye sahip olan adımlar üzerinde durulmalıdır. Plan sonucunda bazı adımların birden fazla defa çalıştırıldığı görülebilir. Bu da yine istenmeyen bir durumdur. Bunların sayısı ne kadar az olursa sorgunun yapılan talebe cevap verme süresi o kadar kısalır [10]. Planlar analiz edilirken dikkat edilmesi gereken en önemli şey sorgunun cevabını alabilmek için tablolardan verilerin alınması sırasında indekslerin nasıl kullanıldığıdır.Eğer grafiksel plan üzerindeki bir tablonun üzerine gelinirse verinin alınması sırasında indekslerin kullanılıp kullanılmadığını, kullanılmışsa nasıl kullanıldığını gösteren bir pencere açılır. Bu mesajlar;

Tablo taraması: Eğer bu mesaj varsa, verinin alınmaya çalışıldığı tablo üzerinde kümelenmiş indeks yoktur ve sonuçlara ulaşabilmek için bir indeksin yardımı alınmamıştır demektir. Eğer bir tablo küçükse tablo taramaları çok hızlı sonuçlanabilir. Hatta böyle durumlarda indeks kullanmak bu hızı azaltabilir.

Dolayısıyla bir tablo taraması yapıldığı görüldüğü zaman yapılması gereken ilk şey; tabloda ne kadar veri olduğuna bakmaktır. Eğer çok fazla veri yoksa tablo taraması performans için en uygun çözümdür. Ama eğer tabloda çok fazla veri varsa o zaman tablo taraması kullanmak performansı olumsuz yönde etkileyecektir.

Ancak şu tarz durumlarla da karşılaşılması mümkündür; uygun bir kümelenmemiş indekse sahip olmasına rağmen sorgu tarafından kullanılmayan ve de tablo taramasıyla sonuçlanan bir sorgu var. Eğer tablodan çekilen veri çok fazlaysa veya aynı kolon değerlerine sahip birçok satır varsa indeks aramak yerine tablo taraması tercih edilir ki bu tercih daha hızlı olacağı için doğru bir tercihtir. Örneğin; 10000 satırlık bir tablo üzerinde bir sorgu çalıştırılıp 1000 satırlık sonuç dönüyorsa bu tablo üzerindeki düğümlenmemiş indeksi kullanılması yerine tablo taraması yapılması daha hızlı olacaktır. Veya 10000 satıra sahip bir tablo olduğu ve bu tablonun 1000 satırının aynı

değere sahip bir kolonu  olunduğu düşünülsün. Bu kolonun sorgunun where ifadesinde yer aldığı düşünülsün. İşte bu durumda da düğümlenmemiş indeks kullanılmasındansa tablo taraması yapılması daha hızlı cevap verecektir.

Çalıştırma planı sonucunda çıkan grafiğe bakılıp da bir tablonun üzerine gelinirse "Hesaplanan Satır Sayısı" diye bir ifadeyle karşılaşılır. Bu sayısı sorgu iyileştiricisinin kaç satır çekileceğine dair yaptığı tahmindir. Eğer bir tablo taraması yapılmışsa ve bu değer çok yüksek çıkmışsa bu normaldir. Optimizasyoncu çok yüksek miktarda veri geleceğini hesaplayarak, düğümlenmemiş indeks kullanılması yerine tablo taramasını tercih etmiş demektir. Bu görüntü yüzünden sorgu üzerinde herhangi bir değişiklik yapmaya gerek yoktur.

-İndeks araması: Çalıştırma planında böyle bir ifadeyle karşılaşılırsa, sorgu iyileştiricisi tablodan verileri almak için düğümlenmemiş indeks kullanmıştır. Bu tarz durumlarda performans oldukça hızlıdır. Özellikle de gelecek satır sayısı azsa.

-Kümelenmiş indeks araması: Eğer çalıştırma planında böyle bir ifadeyle karşılaşılmışsa, sorgu iyileştiricisi, tablo üzerinde kullanabileceği bir kümelenmiş indeks bulmuştur. Kümelenmiş indeksler veriyi almanın en hızlı yoludur.

-Kümelenmiş indeks taraması: Tablo taramasından farkı,kümelenmiş indekse sahip bir tablo üzerinde oluyor olmasıdır. Normal tablo taraması gibi bu da performans sorunlarına sebep olabilir. Bu negatif etki 2 şekilde gerçekleşir. İlk olarak, alınacak veri miktarı tablonun beri miktarına bağlı olarak çok fazla olabilir. Bunun doğrulanması için "Hesaplanan Satır Sayısı" değerine bakılabilir. İkinci olaraksa where ifadesinde olan kolonların sayısı yeterli olmadığından olabilir. Her iki sorunda da kümelenmiş indeks taraması, normal tablo taramasına göre daha hızlı çalışır. Genel olarak kümelenmiş

indeks taramasını, kümelenmiş indeks aramasına çevirmenin yolu daha sınırlayıcı bir sorgu yazılması ve gelecek olan satır sayısının azaltılmasıdır [11].

Eğer çalışılan SQL Server çift işlemciye sahipse ve SQL Serverın varsayılan ayarı olan çift işlemi kullanılabilmesi iptal edilmemişse, sorgu iyileştiricisi bazı sorguların planını oluştururken paralel çalışma ilkesinin var olduğunu varsayarak çalışacaktır. Bu ilkeye göre bir sorgunun çalıştırılması sırasındaki iş yükü birden fazla işlemciye dağıtılabilir. Birçok durumda tek işlemcide çalışan sorgular, çift işlemcide çalışan sorgulardan daha yavaştır.

Sorgu iyileştiricisi potansiyel olarak her zaman paralel çalışmayı kullanabilecek olmasına karşın bunu yapmaz. Bunun sebebi de, sorgu iyileştiricisinin paralellik ilkesini kullanıp kullanmama kararını verirken birden fazla kıstası göz önünde bulundurmasıdır. Örneğin, eş zamanlı kaç aktif bağlantı olduğu, işlemcinin yoğunluğu, sorguları paralel çalıştırabilmek için yeterince bellek olup olmadığı, kaç satırın işleneceği, kullanılacak sorgunun tipinin ne olduğu bu kıstaslardır. Sorgu iyileştiricisi bütün bu verileri toplar ve nasıl çalıştırılacağına karar verir. Aynı sorgu bir dakika önce normal çalıştırılırken aynı sorgu artık paralel çalıştırılabilir.

Bazı durumlardaysa, çoklu işlemci kullanmanın getirdiği yüklenme, sağladığı kaynak kazancından fazladır. Sorgu iyileştiricisi her zaman için artıları ve eksileri düşünmesine rağmen yanılabilir. Bu yüzden eğer paralel çalışmanın sistemi olumsuz etkilendiğinden şüphelenilirse,bazı sorgular için “option(maxdop 1)” ifadesi kullanılarak optimizasyoncunun kararı ezilebilir.

Çalıştırma planlarının grafikleri incelenirken bazı yazıların normalde olduğu gibi siyah fontlarla değil de kırmızı renginde yazıldığı görülebilir. Bunun sebebi ilgili tablonun

istatistiklerinin güncel olmamasından dolayı sorgu iyileştiricisinin daha iyi bir planı ortaya koyamadığını uyarmasıdır. Bunu gidermek için o kırmızı yazının olduğu ikon üzerinde sağ tıklanarak "Eksik Olan İstatistikleri Oluştur" seçeneği seçilir.

Sorgu çalıştırma planları incelenirken sıklıkla "Yer İmi Araması" ifadesiyle karşılaşılır. Temel olarak, sorgu işlemcisinin kümelenmemiş bir indeksten direkt olarak okuması gereken verileri, satır kolonlarına bakarak almaya çalışmasını ifade eder. Örneğin; select, join ve where ifadelerindeki bütün kolonlar kümelenmemiş indeks içerisinde yer almıyorlarsa sorgu iyileştiricisi fazladan iş yüküyle, tablo veya kümelenmiş indekslere bakarak where ifadesinde yer alan bütün kolonları bulmaya çalışır.

Yer imi aramasının bir diğer nedeni de hiç bir zaman kullanılmaması gereken "select * " cümlesinin kullanılmasıdır.

Fazladan I/O gereksinimi olduğu için performans açısından bakıldığı zaman yer imi aramaları tercih edilmemesi gereken durumlardır. Bunlardan kurtulmanın 3 ana yöntemi vardır. Birincisi; where ifadesi tarafından kullanılabilecek bir kümelenmiş indeks oluşturulması.İkincisi; kapsayan kümelenmemiş indeks oluşturulması veya üçüncüsü indeks kesiti kullanılmasıdır.

Sorgu çalıştırma planlarını kullanıp gösterebileceğimiz tek araç sorgu analizcisi değildir. Profiler da bu işin kullanılabilir. Tek farkı grafiksel arayüzü olmamasıdır. Bunun yerine veriler metin verileri halinde sunulmaktadır. Sorguların yakalanması ve profiler'da sorgu çalıştırma planlarının görüntülenmesi için aşağıdaki konfigürasyonu içeren bir çalışma planı yapılmalıdır:

Yakalanması gereken olaylar;

-Performans:Çalıştırma planı

-Performans: Planı Göster-Hepsi

-Performans: Planı Göster-İstatistikler

-Performans: Planı Göster-Metin

Gösterilmesi gereken veri kolonları;

-Başlama zamandı

-Süre

-Metin verisi

-İşlemci

-Okumalar

-Yazmalar

Oluşturulabilecek filtre;

-5 saniye gibi bir filtre verilebilir.

### 3.10. En İyi SQL Server Performans Denetiminin Uygulanması

Bu noktada önerilen bütün önermelerin okunduğu ve/veya uygulandığı varsayılmaktadır. Bazı kısımlarda herkese uymayan şeyler olabilir. Bundan dolayı bu makalede bahsedilen bazı adımlar özelleştirilebilir. Burada yazılanlar genel sorunlara üretilen çözümlerdir.

Denetimleri yaptıktan sonra sonuçlarını bir şekilde takip etmek gerekecektir. Bu tezde verilen kontrol listeleri her bir server için ayrı ayrı uygulanabilir.

Eğer birçok SQL Server ile ve/veya veritabanı ile uğraşılıyorsa bunların performans denetimine ve iyileştirmelere nereden önce başlanacağı belirlenmelidir. İdeal olanı en çok performansa ihtiyaç duyan server ve/veya veritabanlarının denetimin önce yapılmasıdır. Yapılan iş detaylı ve dikkat gerektiren bir iş olduğu için bütün serverlar için ayrı bir dikkat ve vakit ayırımı yapılmalıdır.

Bu denetim yapılırken akılda bulunması gereken şey, yapılanlar SQL Serverların performans sorunlarının tanımlanması ve çözülmesidir. Bu listenin dışındaki sorunlarla karşılaşılması durumunda tezde verilen kontrol listelerine veya adımlara bu yeni sorunlar ve çözümleri de eklenebilir.

Bulunan verilerin toplanması, değerlendirilmesi, çözümlerin üretilmesi ,mümkünse bir test ortamında test edilmesi en son üretim ortamına geçilmesi tavsiye olunur. Karar verilen değişikliklerin canlı ortamı olumsuz yönde etkilemesi gibi olumsuz durumlara hazırlıksız yakalanılmamalıdır.

Yapılacak değişikliklerin hepsi bir anda yapılmamalıdır. En azından ortaya koyduğu değişiklikler gözlenmeden diğer değişikliğe geçilmemelidir. Çünkü ortaya çıkacak bir performans düşüklüğünün hangi adımdan kaynaklandığı veya beklenenden daha düşük performans artışına sebep olan adımın belirlenmesi mümkün kılınmalıdır.

Yapılan değişiklikler bir plan dahilinde ve kontrol listeleri yardımıyla yapıldığı için oluşabilecek ters bir durumu geri çevirmek genellikle kolay olacaktır. Ancak her zaman değil. Bundan dolayı bir B planı her zaman olmalıdır. En azından sistemde değişiklikler

yapılmadan önce en son çalışan sistemin yedeklenmesi oluşabilecek bir tehlikeye karşı alınabilecek en pratik önlemdir.

Zamanla birçok server veri anlamında, donanımların performansı anlamında değişime uğrayacaktır. Bu yüzden server üzerinde bir yıl önce yapılan denetimler sonucunda uygulanan optimizasyon çözümleri 1 sene sonra olumlu cevap vermeyebilir.

# BÖLÜM 4. KONTESKUEL PROGRAMI

Konteskuel programı visual studio program geliştirme arayüzü kullanılarak yazılmış olan bir uygulamadır. Araştırmanın konusunu oluşturan veritabanı MS SQL Server 2000 olduğu için de veritabanı olarak bu veritabanı kullanılmıştır. Uygulamanın amacı, kendisine girdi olarak verilen bir t-sql cümlesini yorumlayıp, daha hızlı sonuç üretebilecek bir sorgu cümlesi olup olmadığını kontrol etmektir.

## 4.1. Programın Kullanılması

Program ilk açılışında şekil 2.1'deki gibi bir görüntüyle karşılaşılır. SQL Server bölümünde sistemdeki varsayılan olarak kurulmuş olan SQL Server 2000'in adı yazmaktadır. Veritabanı bölümü ise bu sunucuda bulunan veritabanlarının isimleriyle otomatik olarak dolar. Kullanıcı adı ve şifresi seçilen veritabanına göre girilir. Eğer bu veriler doğruysa uygulama, seçilen veritabanına ilgili bağlantıyı kurar.

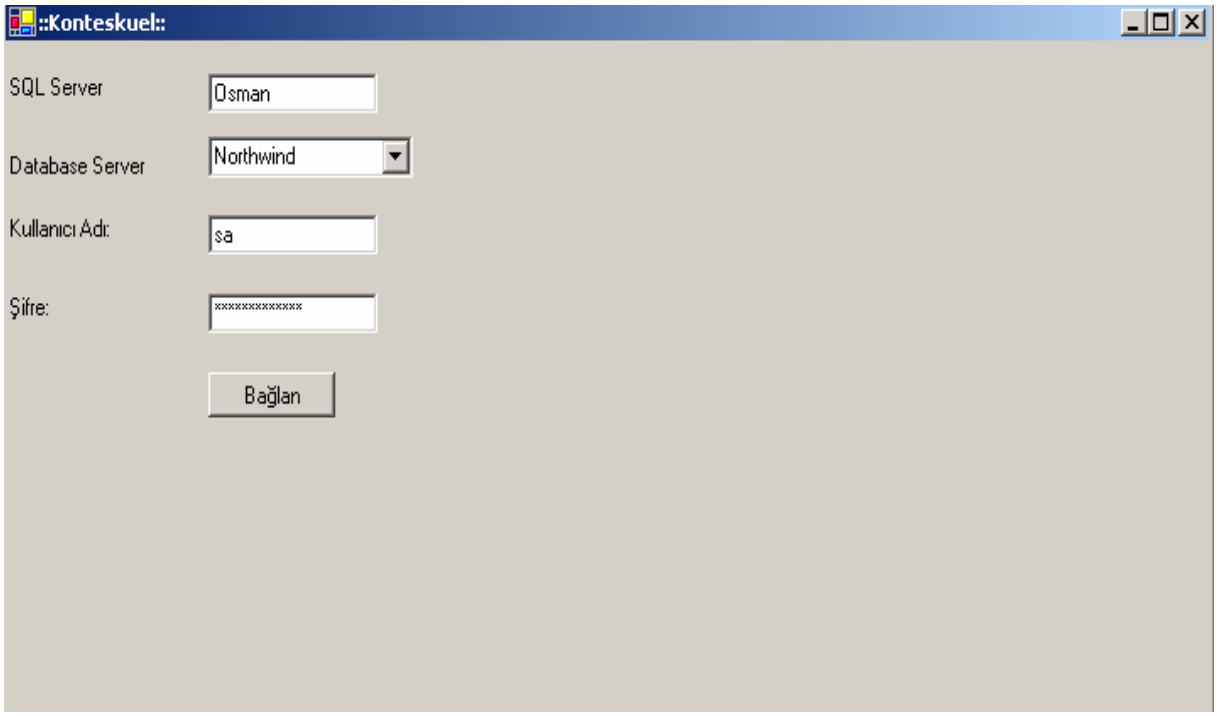
Açılan yeni pencerede sorgu cümlesi bölümüne kontrolü yapılması gereken t-sql cümleciği yazılır. Daha iyisi varsa öner kutucuğu işaretlenirse sorgu düzenleyicisi çalışarak sorgunun daha hızlı sonuç verebilecek bir çalıştırma planına sahip alternatifi var mı diye gerekli algoritmaları çalıştırır. Eğer daha iyi bir alternatifi varsa bunu ekran üzerinde gösterir. Eğer daha optimum bir sorgu önerisi yok ise bunu belirten bir pencere ekranda görülür.

Daha iyisi varsa öner kutucuğu işaretlenmeden yazılan bir sorguyu çalıştırıp ne kadar sürede verilerin ekrana yansıdığı görüntülenebilir. Bu süre ekran üzerindeki süre bölümünde yazar. Daha sonra programdan daha iyi bir sorgu önerisi yapılır. Bu öneri de çalıştırılarak geçen süre

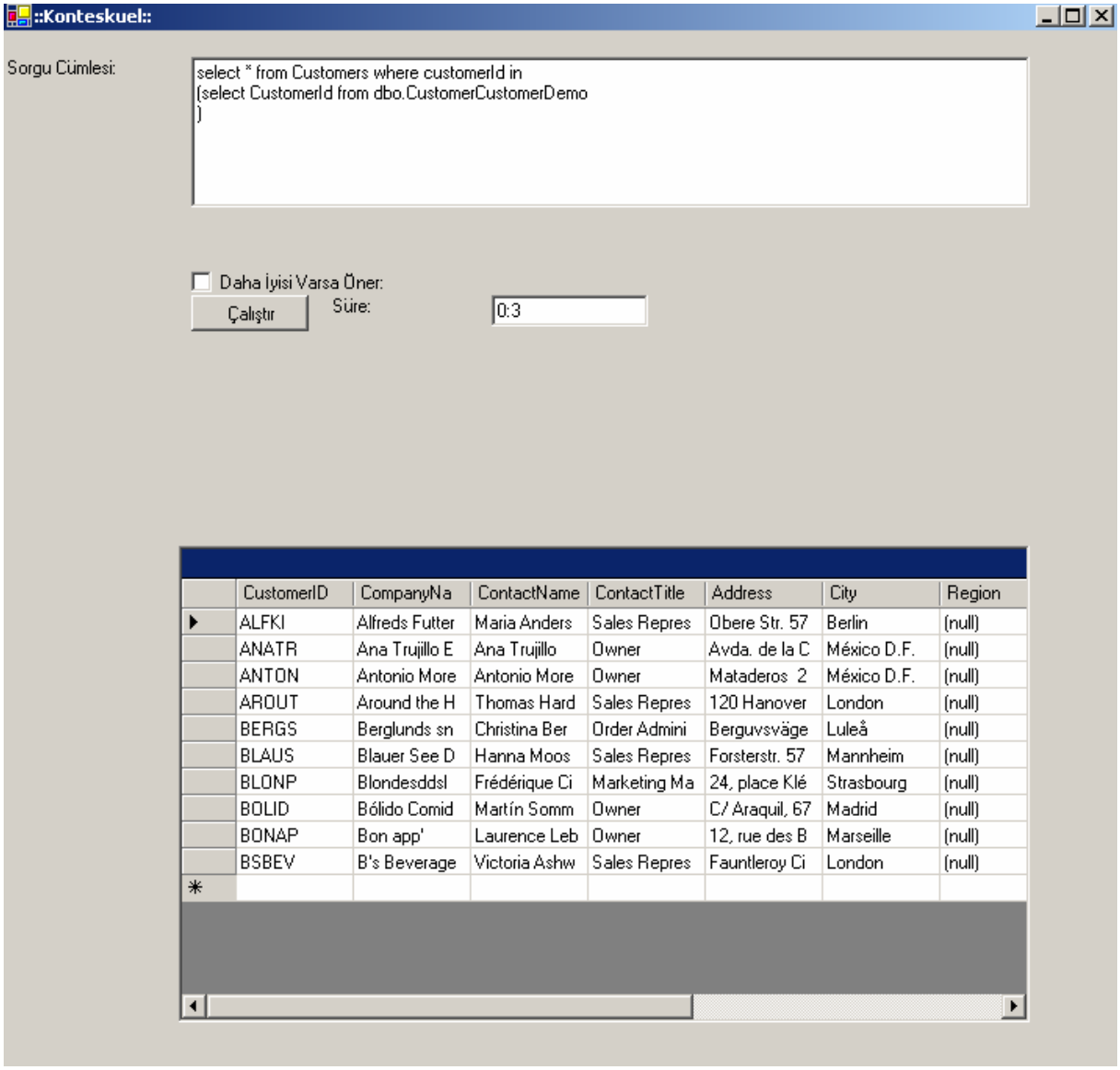
gözlenir. Ortaya çıkan ilk süre ile ikinci süre kıyaslanarak önerilen sorgunun kullanılıp kullanılmayacağı belirlenebilir.

Programın sorguyu optimize eden algoritması araştırmanın 3.7. bölümde anlatılan sorgu optimizasyonu kurallarını temel olarak alır. Aynı zamanda tablolar üzerindeki indeksleri arayıp sorgularda birleştirme işleminde alanlar veya where ifadesine girmiş olan alanlar üzerinde gerekli indeksler olup olmadığını kontrol eder.

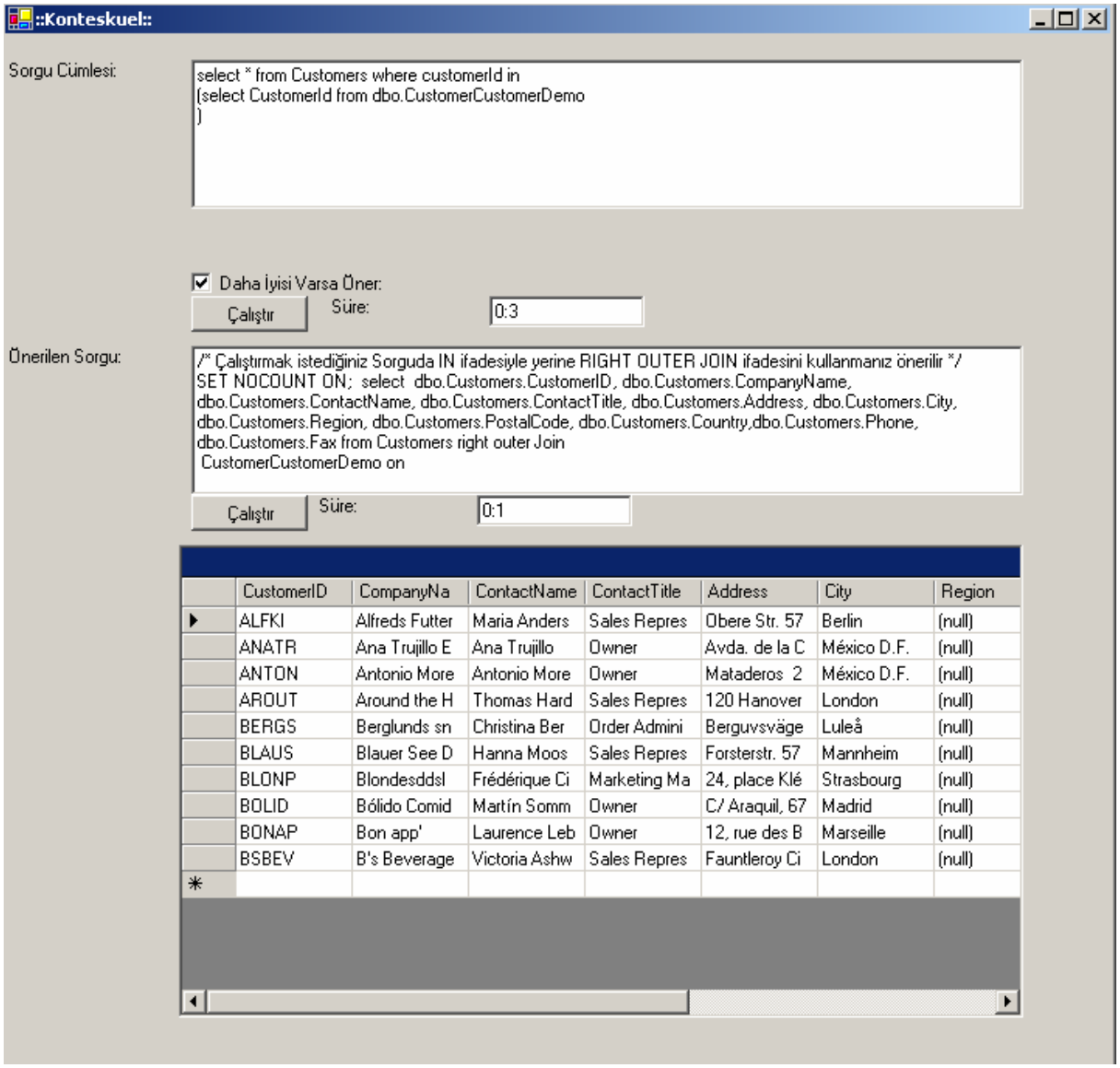
## 4.2 Konteskul Programının Arayüzleri




Şekil 2.1 Programın giriş ekranı




Şekil 2.2 Programa sorgu girişi ve görüntülenmesi




Şekil 2.3 Programın önerdiği sorgu ve çalıştırılması

Programın sürelerinin daha iyi kıyaslanabilmesi için küçük bir Script ile Northwind veritabanındaki Orders tablosunun veri sayısı 600 bine çıkartılmıştır. Programda çalıştırılan bazı sorgular, program tarafından yapılan öneriler, her iki sorgunun çalıştırılma sürelerinin kıyaslaması şöyledir;

"Select * from Orders" sorgusu normal olarak 16 saniyede gelmiştir. Programın bu sorguya alternatif olarak sunduğu sorgu ise;

“SET NOCOUNT ON

Select
OrderID,CustomerID,EmployeeID,OrderDate,RequiredDate,ShippedDate,ShipVia,Freight,

ShipName,ShipAddress,ShipCity,ShipRegion,ShipPostalCode,ShipCountry

from Orders”

sorgusunun ekrana cevap getirme süresi 16 saniyedir.

“Select Distinct OrderID from Orders “ sorgusunun cevabının ekrana herhangi bir optimizasyon yapılmadan önce gelme süresi 4 saniyedir. Bu sorguya programın alternatifi ise;

“SET NOCOUNT ON

select OrderID from orders”

Bu sorgunun sonucu ekrana getirme süresi ise 3 saniyedir.

“Select * from Orders where substring(CustomerID,1,1)='A'” sorgusunun cevabının ekrana gelme süresi 5 saniye iken programın bu sorgu için önerdiği alternatifi;

“SET NOCOUNT ON

Select
OrderID,CustomerID,EmployeeID,OrderDate,RequiredDate,ShippedDate,ShipVia,Freight,

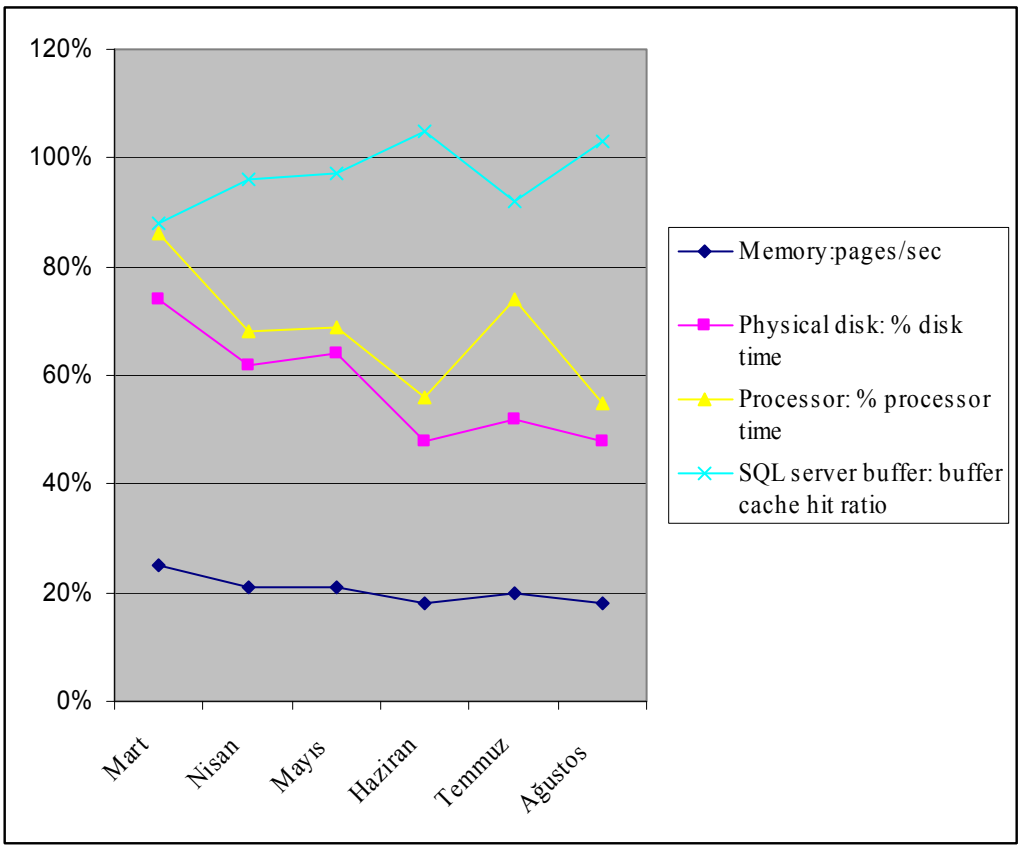
ShipName,ShipAddress,ShipCity,ShipRegion,ShipPostalCode,ShipCountry from Orders where substring(CustomerID,1,1)='A'”

Bu sorgunun yanıtı ekrana döndürme süresi ise 3 saniyedir.

# BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Bu tez araştırması boyunca anlatılan bütün optimizasyon kuralları, standart önerilerden yola çıkılarak, yapılan uygulamalar sonucunda, Server'ın vermiş olduğu tepkiler gözlemlenerek bütün optimizasyon kuralları artısıyla eksisiyle ortaya koyulmuştur. Dış etkenlerden olan SQL Server donanım performansı konusu canlı ortamda test edilememiştir. Bu etkenin, yapılan değişiklikler öncesi ve sonrasındaki yarattığı farklılıkların gözlenebilmesi için gerekli olan operasyonun maddi maliyeti yüksek olduğu için, işlemci değiştirmek, RAM arttırmak, disk dizilerinin tipini değiştirmek gibi değişikliklere gidilememiş ancak yapılması gerekenler madde madde listelenmiştir.

Altı aylık dönemin, üç aylık ilk döneminde SQL Server 2000 bütün mevcut varsayılan değerleriyle konfigüre edilmiş ve performans görüntüleme aracıyla takip edilmiştir. İkinci 3 aylık periyotta ise çalışma boyunca bahsedilen ve önerilen değişiklikler sunucuya uygulanmıştır ve sunucunun hem kaynakları aşırı kullanımının önüne geçilmiş, hem de sorgulara verilen cevaplar hızlanmıştır. Uygulamanın ilk başlarında gelen yavaşlık şikayetlerinin önüne geçilmiştir. Çalışmanın aşamalarının test edildiği makinenin ilgili donanım konfigürasyonu şöyledir; HP Series DL380 G4, Xeon 3.2 Ghz Dual Core CPU, 4 GB RAM, 410 GB data disk, RAID 5 disk dizisi. Sistemin üzerinde koştuğu işletim sistemi ise MS Server 2003'dür. Bu sistem üzerindeki test edilen çalışmanın 6 aylık izleme sonucunda, sistem kaynaklarının kullanımı ve gelen sorgulara verdiği cevapların performansı aşağıdaki grafiklerlerde görülmektedir.




Şekil 2.4 Kaynakların kullanımının zamanla değişim grafiği (1)

Mart 2006'dan Mayıs 2006 sonuna kadar SQL Server 2000'in varsayılan değerleriyle çalışılmıştır. Bu sunucudan beslenen saha otomasyonu projesi, ilaç firmasının saha ekibinin günlük bütün işlemlerini üzerinde yaptığı bir uygulamadır. Mart ve temmuz aylarında, şirkette kampanya dönemleri olmuştur ve tıbbi satış mümessillerinin satış ve ziyaret aktiviteleri artmıştır. Bu önbilgiler eşliğinde grafik incelenecek olursa;

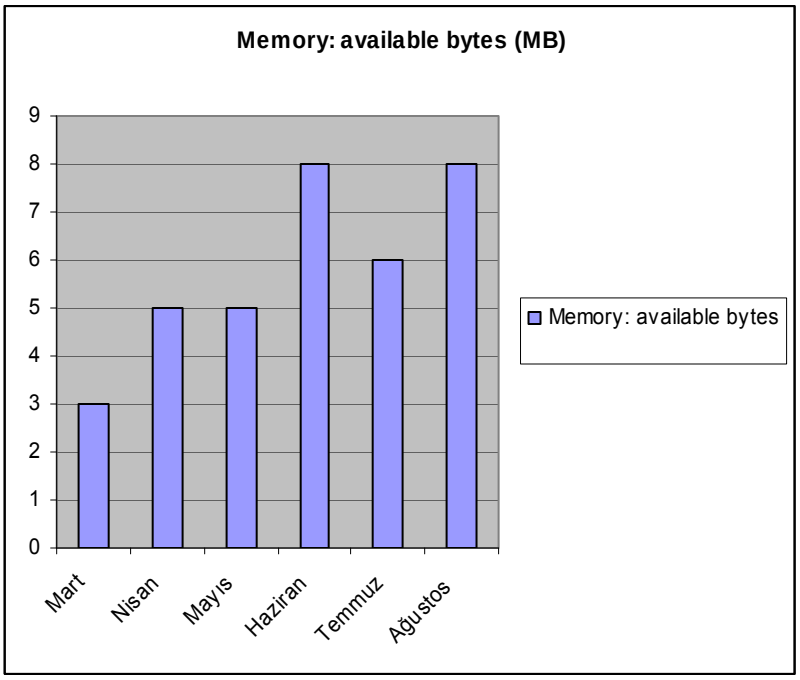
- Memory:pages/sec: Bu sayaç , RAM'den diske saniyede geçen sayfa sayısını belirtir. Ne kadar fazla sayfa geçişi olursa server üzerinde o kadar fazla I/O yükü biner. Bu da performansı olumsuz yönde etkiler. Bu sayacın yüzdesi 0 ile 20 arasında olmalıdır. Buradaki anahtar cümle ortalama sayfanın %20'den az olması gerekliliğidir. Bu oranın %20'den fazla olması RAM ihtiyacını ortaya çıkartır. Grafikte görüldüğü üzere mart-

nisan döneminde bu oran %25 seviyesindedir. Buradan da anlaşılacağı gibi, SQL Server'ın RAM ihtiyacı oldukça yüksek olmuş ve sistemde RAM takviyesi için gerekli veriler ortaya çıkmıştır. Kampanya döneminin olmadığı nisan ve mayıs aylarında bu oran %21'lerde ölçülmüştür. Sistem normal bir şekilde devam ederken bile sisteme RAM takviyesi gereklidir. Ancak bu çalışma boyunca anlatılan denetim çalışmaları sonucunda ortaya çıkan sonuçlar doğrultusunda, haziran, temmuz ve ağustos aylarında yapılan optimizasyon hareketlerinden sonra, daha ilk aydan bu performans ölçütünde %18 seviyesine düşüş gözlenmiştir. Sistem kendisi için gerekli olan RAM ihtiyacını daha verimli bir şekilde kullanmış ve RAM takviyesine olan ihtiyaç ortadan kalkmıştır. Ancak optimizasyon çalışmaları yapılmış olsa da yine bu dönemdeki kampanya çalışmalarından dolayı temmuz ayında bu oran %20 oranına yükselmiştir.

-Physical disk: % disk time: Bu sayacın ölçtüğü değer ise fiziksel bir diskin ne kadar meşgul olup olmadığıdır. Dizilerin ne kadar meşgul olduğunu görmek için güzel bir karşılaştırma verisi verir. Bir kural olarak % Disk Time sayacı %55'den daha küçük olmalıdır. Eğer bu yüzdenin üzerinde bir seviyede bulunuyorsa SQL Server'da bir darboğaz oluşacak denilebilir. Grafikten görüleceği üzere bu değer mart ayı için, kabul edilir değerlerin çok üzerinde ve %74 seviyesinde. Halbuki bir diğer kampanya dönemi olan ve yoğun olarak OLTP işlemlerinin yapıldığı temmuz ayında bu değer yalnızca %52 seviyesinde. Diğer ayların karşılaştırılması da yapıldığı zaman, gerekli optimizasyon çalışmalarının yapıldığı, haziran, temmuz, ağustos dönemlerinde değerlerin kabul edilebilir düzeylere geldiği görülecektir.

-Processor: % processor time: Server'da bulunan bütün işlemciler için kullanılıp veri üretebilir. Ayrı ayrı işlemcileri üretip her biri için ayrı ayrı sonuçlar çıkarabileceği gibi sistemin toplam performansı konusunda da değerler sunar. CPU'yu takip etmek için anahtar sayaç bu sayaçtır. Eğer toplam değer %80'i aşarsa sistemde bir işlemci darboğazı olacağından bahsetmeye başlanabilir. İlk 3 aylık dönemin ortalaması %74 olarak gözlemlenmişken, ikinci 3 aylık dönemde bu ortalama %61 olarak ölçülmüştür.

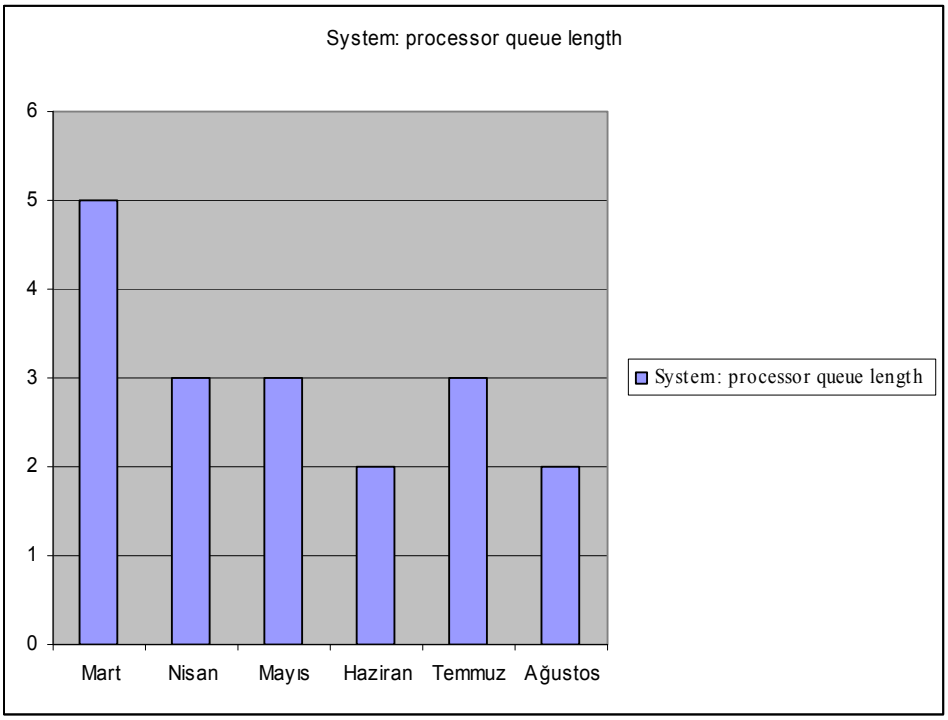
- SQL server buffer: buffer cache hit ratio : Bu sayaç SQL Server'ın veriyi alabilmek için hangi sıklıkla hard diske değil de tampon önbelleğine gittiğini belirtir. OLTP uygulamalarında bu oran %99 ve daha üzeri olmalıdır. Eğer bu oran %90'ın altındaysa derhal RAM alınması zorunludur. Mart ayında bu oran %90'ın bile altındayken ilk 3 aylık dönemin ortalaması 93.6 çıkarak olması gereken %99'luk ortalama değerin altında kalmıştır. Sistemin bellek kaynakları üzerinde darboğazlar oluşmuştur. İkinci 3 aylık dönemde ise bu ortalama %100'de kalmıştır. En düşük seviyesine kampanya dönemi olan temmuz ayında %92 seviyesiyle ulaşmıştır. Ancak ortalama olarak bakıldığı zaman yapılan optimizasyon çalışmaları bu sayaçta da olumlu sonuçlar vermiştir.




Şekil 2.4 Kaynakların kullanımının zamanla değişim grafiği (2)

Şekil 2.4'de değişimi gösterilen sayaç ise memort:available bytes'dır. Bu değer de 5 MB'tan büyük olmalıdır. SQL Server'ın üzerinde koştuğu makinede SQL Server 4–10 MB arasında boş fiziksel bellek arayışındadır. Kalan bellek ise işletim sistemi tarafından

ve SQL Server tarafından harcanır. Eğer kullanılabilir byte miktarın 5MB veya daha az olursa SQL Server'ın bellek yetmezliğinden bir performans sıkıntısı yaşayacağı kesindir. Bu bilgiler doğrultusunda, ortaya çıkan grafik incelenecek olursa, ilk kampanya dönemi olan martta, ortalama 3 MB olarak ölçülmüştür. Bellek ihtiyacı burada kendini göstermiştir. Nisan ve mayıs aylarında ise bu değer sınırda kalmıştır. Tez çalışması boyunca anlatılan bütün optimizasyon maddelerinin uygulanması sonucunda haziran ayında bu değer 8 MB'a yükselerek kabul edilir değerine ulaşmıştır. Sadece temmuz ayında yine kampanya olmasından ve dolayısıyla sunucu aktivitelerinin artmasından dolayı bu değer 6 MB'a düşmüştür. Yine de kabul edilebilir sınır olan 5 MB seviyesinin üzerinde kalmıştır.




Şekil 2.4 Kaynakların kullanımının zamanla değişim grafiği (3)

Bu grafikte ise işlemcide sırada bekleyen thread'lerin ortalama değerleri gözlenmiştir. Eğer işlemci başına sistemde bekleyen işlem sayısı ikiyi geçiyorsa CPU darboğazı ile karşılaşılabilir. Çift işlemciye sahip olan Server'da bu değer 4 sınırında olmalıdır. Ancak görüldüğü üzere yine mart ayında bu değer aşılmış. RAM ihtiyacının yanı sıra CPU darboğazıyla da karşı karşıya kalınmıştır. Nisan ve mayıs aylarında bu değer kabul edilebilir seviyeye gelmiştir. Ancak iyileştirmenin yapıldığı, haziran temmuz ve ağustos dönemlerinin hiç birisinde sınır değer bile gelmemiştir. Yalnızca kampanya dönemi olan temmuzda bu değer, 3'e yükselmiştir ki bu bile kabul edilebilir seviyededir.

Çalışma sonucunda elde edilen iyileştirmeler yalnızca Server'ın kaynaklarının kullanımında olmamıştır. Aynı zamanda t-sql cümleleri ve depolanmış yordamlar üzerinde yapılan değişiklikler sayesinde saha otomasyonu projesinden gelen sorgulara verilen cevap sürelerinde azalmalar gözlenmiştir. Bunun için 6 ay boyunca, sabah 9-11 arasında öğlen 15-17 arasında ve gece 01-03 saatleri arasında Profiler ile veriler toplanmıştır. Profiler'ın izlemesi için Stored Procedures--RPC:Completed , TSQL--SQL:BatchCompleted ana kriterleri verilmiştir. Bu kriterler için, çalıştırılma süresi, yazma işlemi, okuma işlemi, başlama süresi, bitiş süresi verileri gözlenmiştir. Filtre olarak da çalışma süresi 10 saniyeyi geçen sorgular ve depolanmış yordamlar toplanmıştır. Profiler tarafından yakalanan sorguların ilk 3 aylık süre içerinde %12'si bu kritere uymuştur. Mayıs ayı sonrasında depolanmış yordamlar üzerinde ve sorgular üzerinde yapılan, anahtar kelimelerin doğru yerde kullanılması, indekslemenin optimize edilmesi, isimlendirmenin uygun yapılması gibi toplam sorgu süresi optimizasyonu çalışmalarından sonra ( bu çalışmalar 2.6, 2.7, 2.9 bölümlerinde detaylı bir şekilde antalılmıştır.) bu oran % 6'ya inmiştir. Bu çalışmaların yanı sıra bazı spesifik sorguların kontrol edilmesinde ve düzeltilmesinde, araştırma süresinde elde edilen sorgu performansı arttıran kriterler doğrultusunda vb.net koduyla yazmış olduğum "konteskuel" programından faydalanılmıştır.

Tablo 3.10 Önerilen ve ölçülen optimum değerlerin kıyaslanması

| İzlenen Performans Kriteri | En iyi değer olarak beklenen değer | En iyi değer olarak ölçülen değer |
|---|---|---|
| Memeory: Pages/Second | 0-20 arası | Yapılan uygulamalar sonucunda en iyi performans %18'da ölçülmüştür. |
| Physical Disk: % Disk Time | < %55 | Yapılan uygulamalar sonucunda en iyi performans %51'de ölçülmüştür. |
| Proccessor: % Proccessor Time | < %80 | Yapılan uygulamalar sonucunda en iyi performans %64'de ölçülmüştür. |
| SQL server buffer: buffer cache hit ratio | >%99 | Yapılan uygulamalar sonucunda en iyi performans %103'de ölçülmüştür. |
| Memory: available bytes | >5 MB | Yapılan uygulamalar sonucunda en iyi performans 7 MB'da ölçülmüştür. |
| System Proccessor Queue | < 2 (işlemci başına) | Yapılan uygulamalar sonucunda en iyi performans 2 adet işlemde ölçülmüştür. |

Bu çalışma yalnızca SQL Server 2000 için yapılmış ve arada SQL Server 2005 içinde bilgiler verilmiştir. Yapılan diğer çalışmalarla kıyaslandığında [1] [2] [3] [4] [5] [6] jenerik bir veritabanı yönetim sistemlerinde kaynak yönetimi çalışması olmaktan çıkıp, spesifik bir veritabanı yönetim sistemi üzerinde hem kaynak yönetimiyle ilgili hem de sorguların cevap verme sürelerinin kısaltılabilmesi için SQL Server 2000'de performans optimizasyonuna etki edebilecek bütün iç ve dış etkenler incelenmiştir. Benzer konuda çalışma yapılmak istenirse bu çalışmadaki şu eksiklikler giderilmeye çalışılabilir: Çalışmanın tamamı SQL Server 2000 üzerinde yapılmıştır. Ancak araştırma çalışmasına başladığım tarihte SQL Server 2005 henüz piyasada olmadığı için, 2005 üzerinde uygulama ve test yapılmamıştır. MS SQL Server'ın son versiyonu olan 2005 üzerinde bu çalışmalar yapılabilir. Ayrıca maliyet yüksekliğinden ötürü SQL Server donanım performansı denetimi teoride kalmıştır. Burada anlatılanlar pratiğe geçirilerek düzeltmeler, eklemeler yapılabilir. Son olarak vb.net koduyla yazılmış olan konteskuel programının kodu geliştirilebilir. Program girdi olarak verilen bazı sorgulara alternatif sorgu önerisinde bulunamamaktadır. Mümkün olduğunca çok durum göz önüne alınarak verilen sorguların büyük çoğunluğuna alternatif üretmesi sağlanabilir.

## KAYNAKLAR


[1] BERNSTEIN Phil et al. The Asilomar Report on Database Research, SIGMOD Record, Vol. 27, No. 4, pp. 74-80, December 1998.

[2] BROWN Kurt P., MEHTA Manish, CAREY Michale J. and LIVNY Miron. Towards Automated Performance Tuning For Complex Workloads, Proceedings of the 20th International VLDB Conference, pp. 72-84, Santiago, Chile, 1994.

[3] SURAJIT Chaudhuri. Letter from the Special Issue Editor, Bulletin of the Technical Committee on Data Engineering, Vol. 22, No. 2, page 2, June 1999.

[4] MARTIN Patrick, ZHENG Min, HOI-YING Li, ROMANUFA Keri and POWLEY Wendy. Dynamic Reconfiguration: Dynamically Tuning Multiple Buffer
Pools, Proceedings of the International Conference on Database and Expert System Applications (DEXA'2000), pp. 92-101, September, 2000.

[5] WEIKUM Gerhard, HASSE Christof, AXEL Mönkeberg and PETER Zabback. The Comfort Automatic Tuning Project, Information Systems, Vol.
19, No. 5, pages 381-432, 1994.

[6] BENOIT Darcy G., Automatic Diagnosis pf Performance Problems in Database Management Systems, Proceedings of the Second International Conference on Autonomic Computing,2005.

[7] JENNEY L. F., Microsoft SQL Server 2000 Optimization Guide, Printice Hall; 114, 2001.

[8] ENGLAND K., Microsoft SQL Server 2000 Performance Optimization Tuning Handbook , Digital Press, 67, 2001.

[9] WOODY B., Essential SQL Server 2000, Addison-Wesley, 74,2002.

[10] http://www.sql-server-performance.com

[11] http://www.microsoft.com/sql/default.mspx

## Ek-A Çalışma Süresince Kullanılan Teknik Terimlerin Türkçe Karşılıkları

| | |
|---|---|
| Bookmark-lookup | Yer imi araması |
| Bottleneck | Darboğaz |
| Buffer cache | Tampon önbelleği |
| Check point | İşaretleme noktası |
| Clustered index | Kümelenmiş indeksler |
| Compatibility level | Uyum seviyesi |
| Composite-index : | Bileşik indeks |
| Connection pooling: | Bağlantı havuzlama |
| Constraints | Kısıtlar |
| Cursor | İmleç |
| Data warehouse | Veri ambarı |
| Database | Veritabanı |
| Deadloack | Ölümcül kilit |
| Defragment: | Birleştirme |
| Derived tables | Türetilmiş tablo |
| Disk controller | Disk kontrolcüsü |
| Domain controller | Domain kontrolcüsü |
| Execution plan | Çalıştırma planı |
| Fill factor | Doluluk faktörü |
| Foreign key | Yabancı anahtar |
| Fragment: | Bölünme |
| Full table scan | Tam tablo taraması |
| Full Text Search | Tam metin araması |
| Heap | Yığın |
| Hint | İpucu |
| Index Seek | İndeks araması |
| Index tuning wizard | İndeks ayarlama sihirbazı |
| Index view | İndeks kesiti |

| | |
|---|---|
| Identity column | Özdeşlik kolonu |
| İndex | İndeks |
| Job | Görev |
| Join | Birleştirme |
| Joined-tables | Birleştirilen tablolar |
| Leaf page | Yaprak sayfa |
| Log | Kayıt |
| Non-clustered index | Kümelenmemiş indeks |
| Non-clustered index seek | Kümelenmemiş indeks araması |
| Pagesplit | Sayfa bölünmesi |
| Paging | Sayfalama |
| Primary key | Birincil ahantar |
| Query analyzer | Sorgu analizcisi |
| Query optimizer | Sorgu optimizasyoncusu/düzenleyicisi |
| Query tuning | Sorgu ayarlaması |
| Referential integrity: | Gösterimle ilgili bütünlük |
| Service pack | Servis paketi |
| Stored procedure | Depolanmış yordam |
| Swap file | Takas dosyası |
| Table scan | Tablo taraması |
| Transaction | İşlem |
| Trigger | Tetikleyici |
| View | Görüntü tablosu |

# ÖZGEÇMİŞ

Osman Ayhan, 05.10.1981'de Seydişehir'de doğdu. İlk, orta ve lise 2'ye kadar olan eğitimini Seydişehir'de tamamladı ve 1999 yılında Karaman Anadolu Lisesi'nden mezun oldu. 1999-2000 yıllarında başladığı Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nden 2003 yılında mezun oldu. 2003-2006 yılları arasında Sandoz İlaç'da yazılım uzman yardımcısı olarak çalışmaya başladı. Bu süreçler esnasında şirketin MIS raporlama sistemlerinde, saha otomasyonu projesinde, satış ve pazarlama departmanına yönelik diğer projelerde görev almıştır. Ocak 2006'dan beri Abdi İbrahim İlaç A.Ş'de yazılım uzmanı olarak çalışmaya devam etmektedir.