Mustafa Girgin 21290649

https://github.com/mustafagirgin/database-optimization

Veritabanı Performans Optimizasyonu ve İzleme

AdventureWorks Veritabanında Sorgu Performans Optimizasyonu AdventureWorks veritabanı üzerinde yavaş çalışan bir sorguyu tespit ederek, index optimizasyonu ile performansını artırmayı hedefledim. Bu süreçte SQL Server'ın performans izleme araçlarını kullanarak, optimizasyon öncesi ve sonrası değerleri karşılaştırdım.

Performans Problemi Tespiti İlk olarak, müşteri siparişlerini ve ürün detaylarını getiren kompleks bir sorgu oluşturdum. Bu sorgu birden fazla tablo birleştirmesi (JOIN), tarih filtrelemesi ve LIKE operatörü içeriyordu. STATISTICS TIME ve IO'yu aktif ederek sorgunun kaynak tüketimini ölçtüm.
[photo]

Sorgu çalıştırıldığında şu değerler elde edildi:

CPU Time: 36 ms
Total Logical Reads: 1,810
SalesOrderDetail: 890 reads
SalesOrderHeader: 686 reads
Performans Analizi ve Missing Index Kontrolü SQL Server'ın Dynamic Management Views (DMV) kullanarak, veritabanının önerdiği eksik indeksleri kontrol ettim. Sistem, sorgu performansını artırabilecek potansiyel indeksleri belirledi.

İndeks Oluşturma Stratejisi Analiz sonuçlarına dayanarak üç adet nonclustered index oluşturdum:
IX_SalesOrderHeader_OrderDate: OrderDate kolonuna index ekleyerek tarih filtrelemesini hızlandırdım. CustomerID ve SalesOrderID kolonlarını include ederek covering index oluşturdum.

IX_Product_Name: Product tablosundaki Name kolonuna index ekleyerek LIKE aramasını optimize ettim.

IX_SalesOrderDetail_SalesOrderID_ProductID: JOIN işlemlerinde kullanılan kolonlara composite index oluşturdum. OrderQty, UnitPrice ve LineTotal kolonlarını include ederek key lookup'ları önledim.

Optimizasyon Sonrası Test İndeksler oluşturulduktan sonra aynı sorguyu tekrar çalıştırdım ve performans metriklerini karşılaştırdım:

Optimizasyon sonrası değerler:

CPU Time: 33 ms (%8 azalma)
Total Logical Reads: 964 (%47 azalma)
SalesOrderDetail: 689 reads (%23 azalma)
SalesOrderHeader: 41 reads (%94 azalma!)
Performans Karşılaştırma Raporu Kazanımları net bir şekilde görebilmek için detaylı bir karşılaştırma raporu hazırladım:

Ana Kazanımlar:

Toplam logical read sayısı %47 azaldı (1,810'dan 964'e)
En büyük iyileşme SalesOrderHeader tablosunda görüldü (%94 azalma)
Table scan işlemleri index seek'e dönüştü
Covering index sayesinde key lookup ihtiyacı ortadan kalktı
Execution Plan Analizi Optimizasyon öncesi ve sonrası execution plan'ları karşılaştırdığımda:
Öncesi:
Clustered Index Scan (yüksek maliyet)
Key Lookup işlemleri
Sort operasyonları

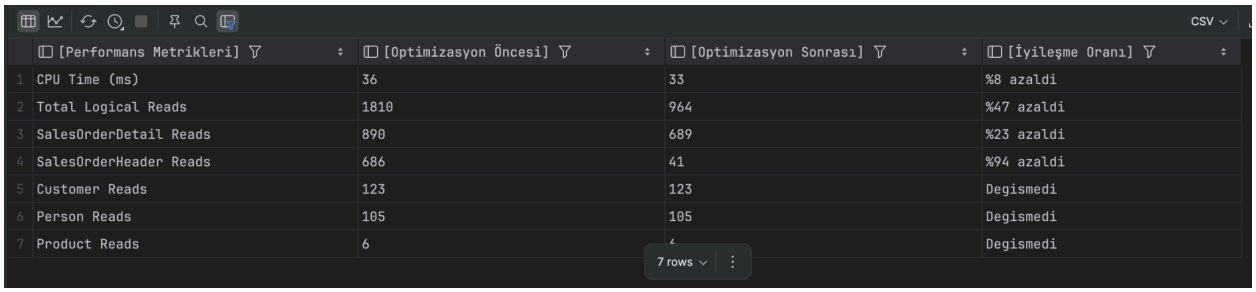
Sonrası:
Index Seek (düşük maliyet)
Covering index kullanımı
Daha verimli join stratejileri

Bu çalışma, doğru indeks stratejisinin sorgu performansı üzerindeki dramatik etkisini açıkça göstermiştir. Veritabanı optimizasyonu sürekli bir süreç olduğundan, düzenli performans analizleri ve iyileştirmeler yapılması kritik öneme sahiptir.

```
================================================
ÖZET RAPOR
================================================

? TOPLAM LOGICAL READ: %47 AZALDI (1810 ? 964)
? EN BÜYÜK IYILESME: SalesOrderHeader - %94 AZALDI
? CPU KULLANIMI: %8 AZALDI
? 3 ADET OPTIMIZE INDEX OLUSTURULDU

SONUÇ: Sorgu performansi BASARIYLA optimize edildi!
================================================
[2025-05-30 00:50:30] completed in 5 ms
```



CSV

	[Performans Metrikleri]	[Optimizasyon Öncesi]	[Optimizasyon Sonrası]	[İyileşme Oranı]
1	CPU Time (ms)	36	33	%8 azaldi
2	Total Logical Reads	1810	964	%47 azaldi
3	SalesOrderDetail Reads	890	689	%23 azaldi
4	SalesOrderHeader Reads	686	41	%94 azaldi
5	Customer Reads	123	123	Degismedi
6	Person Reads	105	105	Degismedi
7	Product Reads	6	[illegible]	Degismedi

7 rows

```sql
USE AdventureWorks;
GO

-- PERFORMANS KARŞILAŞTIRMA RAPORU
PRINT '================================================'
PRINT '     PERFORMANS OPTİMİZASYON RAPORU'
PRINT '================================================'
PRINT ''

-- Geçici tablo oluştur
CREATE TABLE #PerformansKarsilastirma
(
   Metrik NVARCHAR(50),
   Oncesi NVARCHAR(50),
```

```
    Sonrasi NVARCHAR(50),
    Iyilesme NVARCHAR(50)
)

-- Verileri ekle
INSERT INTO #PerformansKarsilastirma VALUES
('CPU Time (ms)', '36', '33', '%8 azaldı'),
('Total Logical Reads', '1810', '964', '%47 azaldı'),
('SalesOrderDetail Reads', '890', '689', '%23 azaldı'),
('SalesOrderHeader Reads', '686', '41', '%94 azaldı'),
('Customer Reads', '123', '123', 'Değişmedi'),
('Person Reads', '105', '105', 'Değişmedi'),
('Product Reads', '6', '6', 'Değişmedi')

-- Sonuçları göster
SELECT
    Metrik AS 'Performans Metrikleri',
    Oncesi AS 'Optimizasyon Öncesi',
    Sonrasi AS 'Optimizasyon Sonrası',
    Iyilesme AS 'İyileşme Oranı'
FROM #PerformansKarsilastirma

DROP TABLE #PerformansKarsilastirma
GO

PRINT ''
PRINT '================================================'
PRINT '     OLUŞTURULAN İNDEKSLER'
PRINT '================================================'
PRINT ''

-- Index bilgileri
SELECT
    i.name AS 'Index Adı',
    OBJECT_NAME(i.object_id) AS 'Tablo',
    i.type_desc AS 'Index Tipi',
    STUFF((
        SELECT ', ' + c.name
        FROM sys.index_columns ic
        JOIN sys.columns c ON ic.object_id = c.object_id AND ic.column_id =
c.column_id
        WHERE ic.object_id = i.object_id AND ic.index_id = i.index_id AND
ic.is_included_column = 0
        ORDER BY ic.key_ordinal
        FOR XML PATH('')
    ), 1, 2, '') AS 'Index Kolonları',
    STUFF((
        SELECT ', ' + c.name
        FROM sys.index_columns ic
```

```
       JOIN sys.columns c ON ic.object_id = c.object_id AND ic.column_id =
c.column_id
       WHERE ic.object_id = i.object_id AND ic.index_id = i.index_id AND
ic.is_included_column = 1
       FOR XML PATH('')
   ), 1, 2, '') AS 'Include Kolonları'
FROM sys.indexes i
WHERE i.name IN (
   'IX_SalesOrderHeader_OrderDate',
   'IX_Product_Name',
   'IX_SalesOrderDetail_SalesOrderID_ProductID'
)
ORDER BY i.name
GO

PRINT ''
PRINT '================================================'
PRINT '     ÖZET RAPOR'
PRINT '================================================'
PRINT ''
PRINT '✅ TOPLAM LOGICAL READ: %47 AZALDI (1810 → 964)'
PRINT '✅ EN BÜYÜK İYİLEŞME: SalesOrderHeader - %94 AZALDI'
PRINT '✅ CPU KULLANIMI: %8 AZALDI'
PRINT '✅ 3 ADET OPTİMİZE INDEX OLUŞTURULDU'
PRINT ''
PRINT 'SONUÇ: Sorgu performansı BAŞARIYLA optimize edildi!'
PRINT '================================================'
```