SQL Optimizasyonu ve İndex Kavramı
SQL Optimization and Index Concept

ÖMER DİNER
omer.diner@std.yildiz.edu.tr
20011017

I. VERİTABANINDA PERFORMANS

Etkili veritabanı yönetimi veritabanının çalışma ve taleplere cevap verme performansını en üst seviyede tutmak üzerine kurgulanmalıdır. Performans optimizasyonu girişimleri söz konusu olduğunda, bu bağlamda iki temel unsur öne çıkar: istemci ve sunucu arasındaki veri trafiğini azaltmak ve disk erişimini en aza indirmek.Veritabanı sistemlerini verimli bir şekilde yönetmeye ve kullanıcı deneyimini iyileştirmeye yönelik her yaklaşımın temelini bu iki unsur oluşturur.
SQL sorgularında dikkat edilmesi gereken durumları ve veritabanında indeksleme fikrini anlamak, veritabanı yöneticilerinin ve geliştiricilerinin bu alanlara odaklanarak bu iki önemli bileşeni daha verimli bir şekilde ele almalarına yardımcı olacaktır. Performans odaklı çözümler doğru bir şekilde uygulandığında, veritabanı sisteminin verimliliğini ve kullanılabilirliğini artırabilir aynı zamanda bu kavramlardan habersiz kullanıcıların kullandıkları uygulamalarda sorun yaşamamasını sağlar.

II. SQL YAZARKEN DİKKAT EDİLMESİ GEREKENLER

1. Toplu veri işlemlerinde uygulamayla veritabanı arasında trafiği azaltmak için gerekli ayarlamaları uygulama tarafında yapıp, veritabanına sık sık komut atmaktan kaçınılmalı bunun yerine mümkünse bir komutla birden fazla ekleme, güncelleme yapılması sağlanmalı.

2. SQL sorgularında, "SELECT * FROM table;" kullanımında görüldüğü üzere “*” yani tüm sütunların istenmesi kullanımından mümkün olduğunca kaçınılmalıdır. Bu tür geniş çaplı sorgular, gereksiz veri transferine neden olabilir ve veritabanı performansını olumsuz yönde etkileyebilir. Bunun yerine, sadece ihtiyaç duyulan sütunlar belirtilmeli ve sorgular bu belirtilen sütunlar üzerinden yürütülmelidir. Veritabanı tablolarındaki geniş veri kümesinden yalnızca gerekli olanı seçmek, hem sorgu yanıt sürelerini iyileştirir hem de ağ trafiğini azaltarak daha etkili bir veritabanı yönetimi sağlar.

3. Veri tiplerinin doğru seçimi, verilerin etkili bir şekilde depolanmasını sağlayarak disk alanının verimli kullanımına katkı sağlar. Doğru veri tipi seçimi, hem tablo boyutunu azaltır hem de disk I/O işlemlerini optimize eder. Kullanılan veri tipinin boyutu, depolanan verinin karakteristiğine uygun olarak belirlenmelidir. Örneğin, yıl değeri tahmin edilebilen bir sayı aralığında olduğu için, INT veri tipi bu durumda gereksiz büyük bir bellek alanı kullanımına neden olabilir. Bu durum özelinde SMALLINT veya TINYINT, sadece gerekli alanı kullanarak bellek tasarrufu sağlar.

4. LIKE operatörünün kullanımı, sorgu performansı açısından yavaşlatıcı bir etkendir ve mümkünse bu operatörden kaçınılmalıdır. Ancak, zorunlu durumlarda kullanılması gerekiyorsa, sonuç kümesini daraltacak şekilde kullanmak önemlidir. Örneğin, "SELECT * FROM City WHERE postalCode LIKE '%3256%';" sorgusu yerine, "SELECT * FROM City WHERE postalCode LIKE '3256%';" kullanmak daha performanslı bir seçenektir. Wildcard (%) ifadesinin sorgu sonlarında belirlenmiş olması, veritabanı indekslerinin kullanılmasını sağlar ve bu da sorgu performansını artırır.

5. Uygulama içerisinde SQL kodu yazmak yerine stored procedureler(derlenmiş yordam) kullanılmalı. Bu yordamlar, bir kez derlendikten sonra veritabanında saklanır. Bu işlem, yordam çağrılarının hızlı ve verimli bir şekilde gerçekleşmesini sağlar. Derlenmiş bir yordam, her çağrıldığında tekrar derlenmek zorunda kalmaz, bu da işlemlerin daha hızlı yürütülmesine olanak tanır. Yordamlar yürütülebilir kodu içerir ve bu kod otomatik olarak önbelleğe alınır. Bu durum, sıkça kullanılan yordamların daha hızlı yanıt vermesine olanak tanır. Ayrıca, önbelleğe alma, bellek gereksinimlerini azaltır çünkü sıkça kullanılan yordamların kodu tekrar tekrar belleğe

alınmaz, bu da genel bellek yönetimini optimize eder. Yordamlar, veritabanı yönetim sistemleri tarafından optimize edilebilir. Bu, veritabanının sorguyu daha etkili bir şekilde yürütmek için gerekli değişiklikleri yapmasını sağlar. Ayrıca performans katkısının yanında yordam kullanımı, veritabanı güvenliğini artırır. Kullanıcılara, doğrudan tablolara veya sorgulara erişim izni vermek yerine, yordamları kullanma yetkisi vererek daha sıkı bir kontrol sağlanabilir.

6. Partition olarak adlandırılan verilerin parçalanması kavramı, büyük veri tablolarında işlemleri optimize ederek I/O, CPU ve bellek tüketimini minimize eder. Bu sayede daha büyük veri setlerini küçük parçalara bölerek daha etkili bir şekilde çalışmak mümkün hale gelir. Aynı zamanda, mevcut sorguları değiştirmeden ölçeklenebilir ve sürdürülebilir bir mimariye olanak tanır. Bu kazanımlar, veritabanı performansını artırırken mevcut altyapıda önemli değişikliklere gitme ihtiyacını ortadan kaldırır.

III. INDEX KAVRAMI

Bir index, farklı bellek içi ve disk içi veri yapıları kullanarak arama anahtarlarını diskteki ilgili verilerle eşleştirir. Index, aranacak kayıt sayısını azaltarak aramayı hızlandırmak için kullanılır. Veritabınında saklanan verilerin sayısı arttıkça performansta düşüşler başlar, bazı sorguların çalışması dakikalar hatta saatler alabilir. Dağınık bir yapıda olan verilerde istenilen veriyi aramak için tüm tablonun taranması işlemi yapılır. Bu işlemi küçük boyutlu bir tabloda yapmak kolaydır. Artan veri miktarına göre ise bu işlem vakit kaybettirir. Veriye erişim hızını arttırmak için indexleme yöntemi kullanılır.
Bir index, bir tablodan seçilen veri sütunlarının hızlı arama yapılmasını sağlamak üzere tasarlanmış bir kopyasıdır. Bir index temelde bir "anahtar" ve o anahtarın karşılık geldiği asıl satırın tutulduğu fiziksel veriye erişim için gereken pointerı içerir.
İndexler, veriyi fiziksel olarak yeniden düzenlemek yerine, bir çeşit veri yapısı kullanarak erişim hızını artırır.

Ancak her durumda indexleme kullanmak doğru bir yaklaşım değildir. İndexler, okuma içeren sorgu performansını artırırken diğer yandan yazma işlemlerini yavaşlatabilir. Yeni veri eklemek veya varolan veriyi güncellemek gibi işlemler, indexlerin devamlı güncellenmesine neden olur ve bu da performansı olumsuz etkileyebilir.

Veritabanı performansını optimize etmek adına, okuma işlemlerinin yoğun olduğu tüm tablolarda indexlerin oluşturulması önemlidir. İndexler, sorguların daha hızlı çalışmasını sağlar ve özellikle okuma işlemlerinde performans artışına katkıda bulunur.

A. *İndex Kullanımı Üzerine Tavsiyeler*

- Genel olarak az arama ve okuma işlemi, ancak daha fazla ekleme, silme ve güncelleme işlemi yapılan tablolarda gereksiz indexlerden kaçınılmalıdır. Yazma işlemi yoğun bir tabloda çok sayıda sütun üzerinde index bulunması komutların çalışma hızını önemli ölçüde azaltacaktır, çünkü indexler her işlemde güncellenmesi gereken veri yapılarıdır.

- Tam tablo taraması (Full table scan), sorgunun WHERE kısmındaki sütunların kendileriyle ilişkili bir indeks olmadığında gerçekleşir. Bu durumu önlemek ve sorguların daha hızlı çalışmasını sağlamak için, sorgunun WHERE kısmında kullanılan sütunlar için uygun indexler oluşturulmalıdır. Bu, tam tablo taramasını engeller ve sorguların daha verimli bir şekilde çalışmasına olanak tanır.

KAYNAKLAR

[1] https://www.baeldung.com/cs/databases-indexing
[2] https://www.quora.com/What-is-the-time-complexity-of-a-search-query-in-a-database
[3] https://medium.com/@cankaya07/sql-server-table-partitioning-29a5e490a5ff
[4] *https://en.wikipedia.org/wiki/Database_index*
[5] *https://blog.devart.com/how-to-optimize-sql-query.html*
[6] *https://medium.com/@arifozanakta/sqlde-i%CC%87ndeksleme-sorgu-performans%C4%B1n%C4%B1-art%C4%B1rma-rehberi-566b3847eb8c*
[7] *https://www.sqlekibi.com/sql-server/sql-server-performansini-ve-veritabaninizi-iyilestirmeye-yonelik-ipuclari-%F0%9F%92%A1.html/*