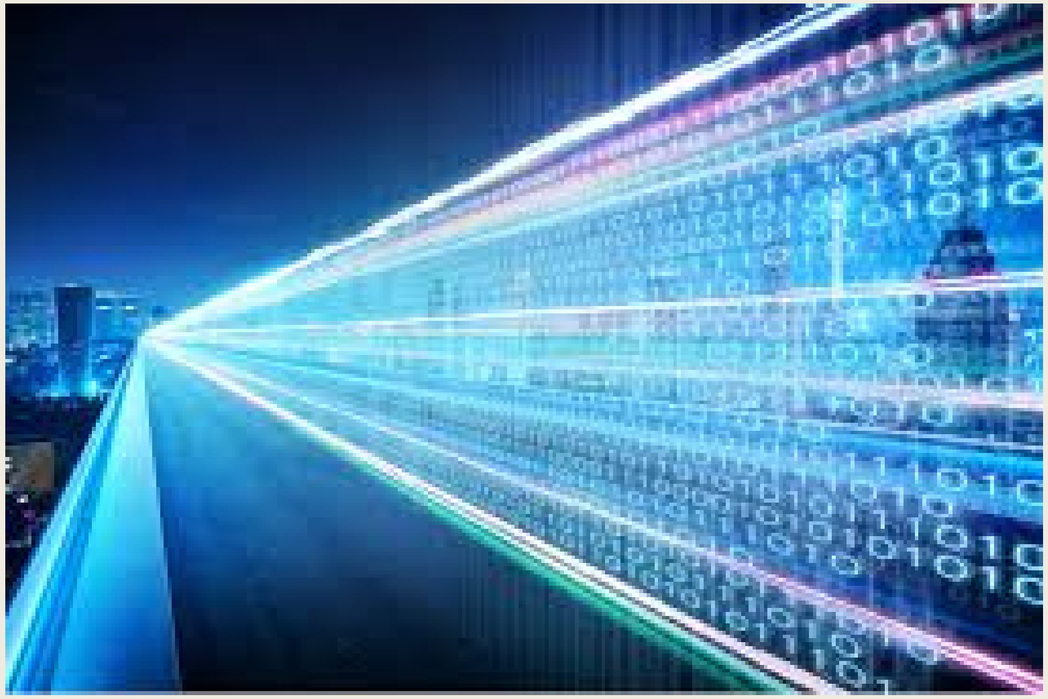
SQL PERFORMANS ARTIRMA VE INDEX KAVRAMI

Ömer Diner 20011017

Performans için bilinmesi gereken iki kritik nokta; **istemci ve sunucu arasındaki veri trafiğinin minimumda tutulması** ve olabildiğince **az disk erişimi** yapılması gerekliliğidir.



TOPLU VERİ İŞLEMLERİ

Var olan bin adet satırı Test adındaki tabloya ekleyelim. Birinci ve **YANLIŞ** olan yöntem **döngü içerisinde satır sayısı kadar INSERT komutu** çalıştırmaktır.

```
Stopwatch stopwatch = new Stopwatch();
stopwatch.Start();
try {
    List<int> değerler = new List<int>();
    using (var connection = new SqlConnection("................")) {
        connection.Open();
        for (int i = 0; i < 1000; i++) {
            using (SqlCommand cmd = new SqlCommand("INSERT INTO Test (Test) VALUES(" + i + ")", connectio
                cmd.ExecuteNonQuery();
            }
        }
    }
} finally {
    stopwatch.Stop();
}
Console.WriteLine("Süre: {0}", stopwatch.Elapsed);
// Süre: 00:00:00.2238716
```

İkinci ve **DOĞRU** olan yöntem ise gerekli parametreleri bir değişkende toplayarak tek bir insert komutu göndermektir.

```
Stopwatch stopwatch = new Stopwatch();
stopwatch.Start();
try {
    List<string> değerler = new List<string>();
    using (var connection = new SqlConnection("................")) {
        connection.Open();
        for (int i = 0; i < 1000; i++) {
            değerler.Add(string.Format("({0})", i));
        }
        using (SqlCommand cmd = new SqlCommand(string.Format("INSERT INTO Test (Test) VALUES {0}", string.J
            cmd.ExecuteNonQuery();
        }
    }
} finally {
    stopwatch.Stop();
}
Console.WriteLine("Süre: {0}", stopwatch.Elapsed);
// Süre: 00:00:00.0478979
```

SELECT * FROM CUSTOMERS; 

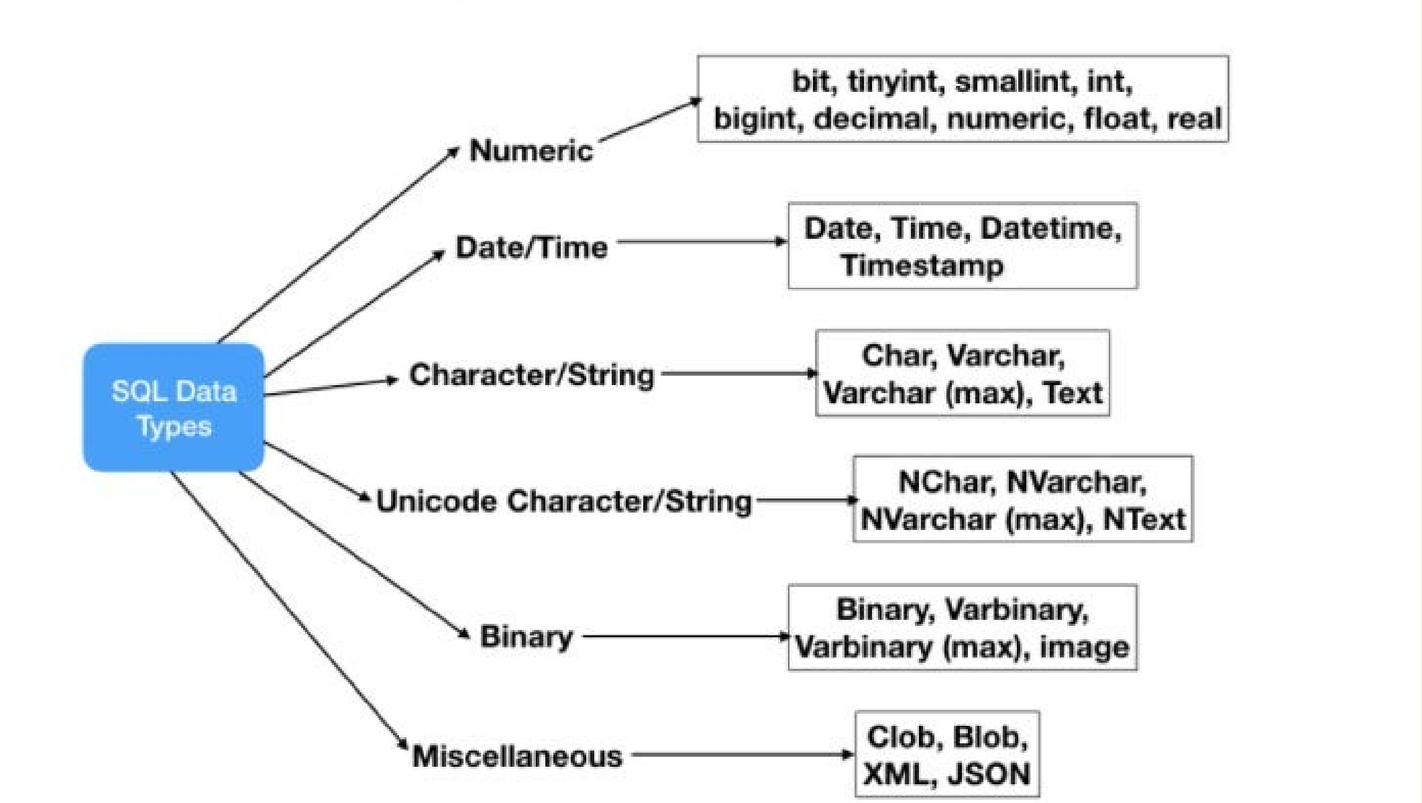
SELECT id,name FROM CUSTOMERS; 

Sadece ihtiyaç duyulan sütunları seçin

UYGUN VERİ TİPİ SEÇİMİ

Verilerin doğru veri tipleriyle depolanması halinde daha az disk alanı kullanılır.

Örneğin,yıl değerini sayısal olarak depolamak için INT veri tipi yerine SMALLINT veya TINYINT veri tiplerini kullanmak daha az bellek kullanımı sağlar. Böylece tablo boyutu azalır ve disk I/O işlemleri optimize edilir.



LIKE KULLANIMI

Like kullanımından mümkün olduğunca kaçınılmalıdır. Eğer mutlaka kullanımı gerekiyorsa sonuç setini daraltacak şekilde kullanılmalıdır.

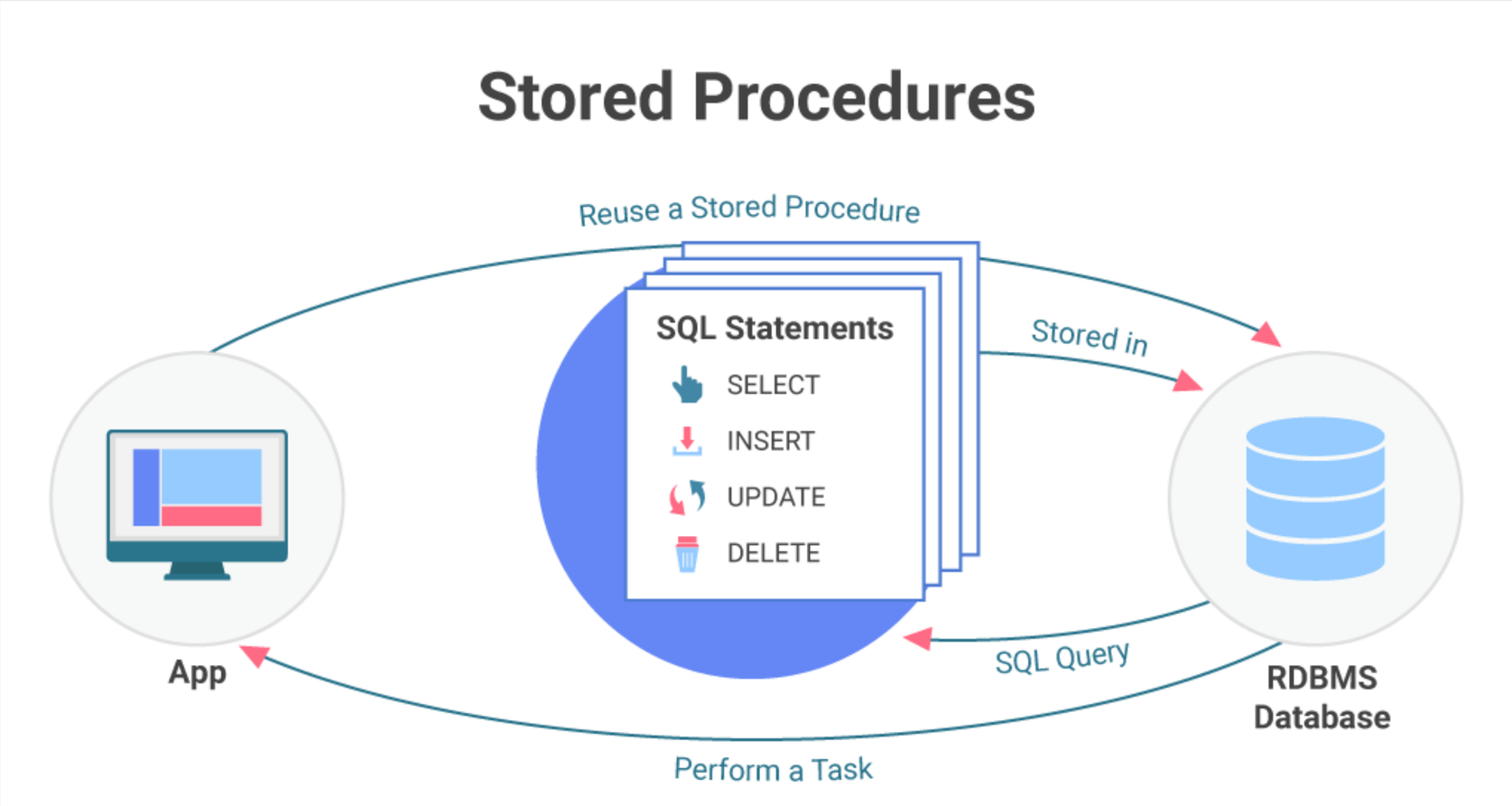
SELECT * FROM City WHERE postalCode LIKE '%3256%'; yerine

SELECT * FROM City WHERE postalCode LIKE '3256%';

daha performanslı bir sorgudur. Wildcardın(%) sonlarda belirlenmiş olması varsa index kullanımını sağlar dolayısıyla performansı arttırır.

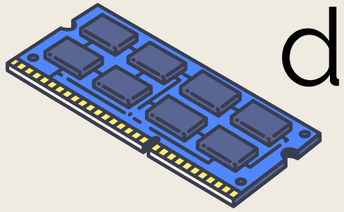
PROCEDURE KULLANIMI

Sık kullanılan veriler ve daha karmaşık sorgular için stored procedure kullanmak veritabanı performansını olumlu yönde etkiler. Bunun birkaç nedeni vardır:



Yordamlar **bir kez derlenip sonrasında çalıştırılabilir biçimde saklandığından**, yordam çağrıları hızlı ve verimlidir.

Yürütülebilir kod otomatik olarak **önbelleğe alınır**, dolayısıyla bellek gereksinimleri azalır.

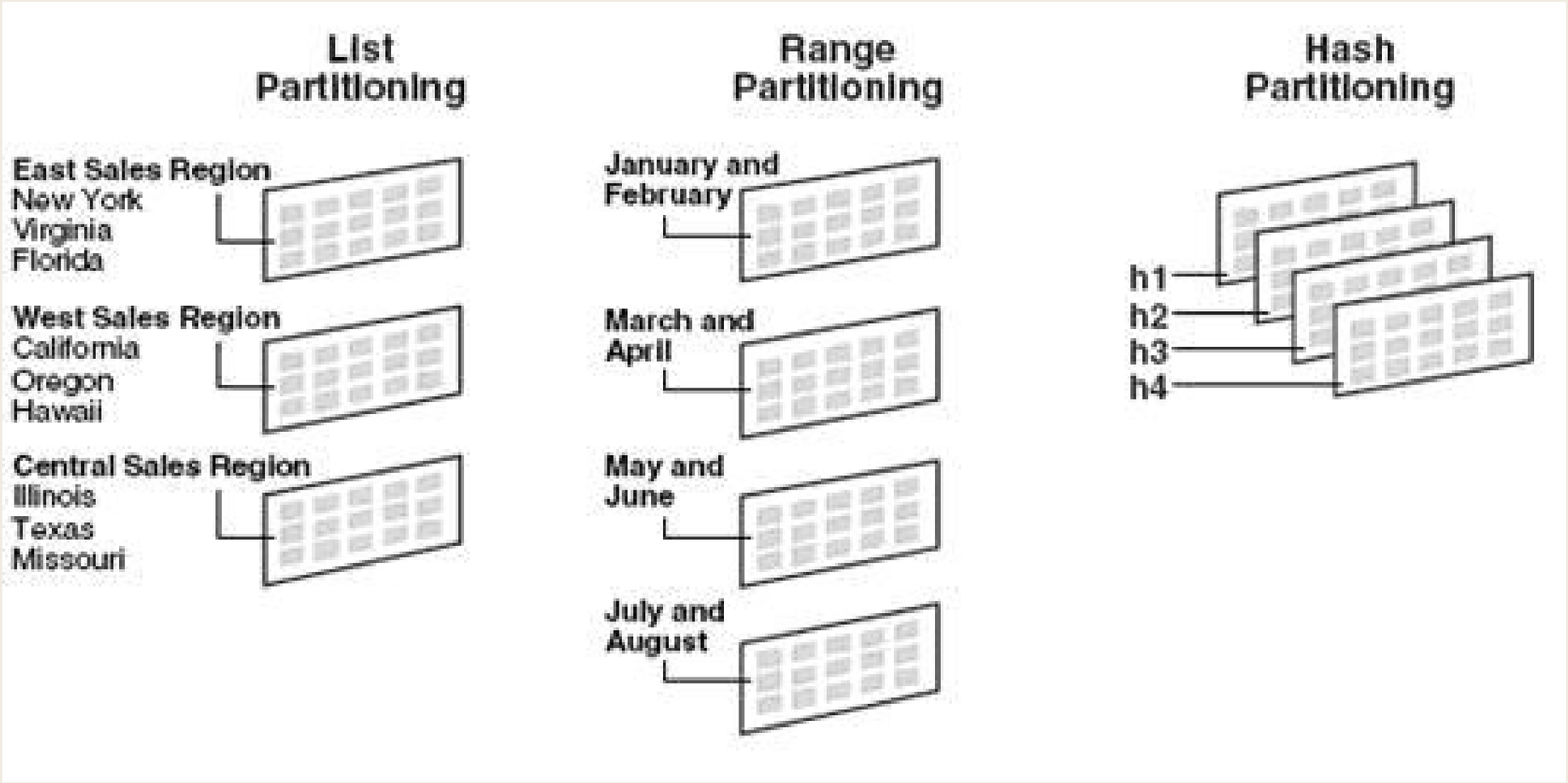


Yordamlar, veritabanı tarafından **optimize** edilebilir. Bu, sorgunun daha verimli bir şekilde yürütülmesi için veritabanı tarafından gerekli değişikliklerin yapılması anlamına gelir.

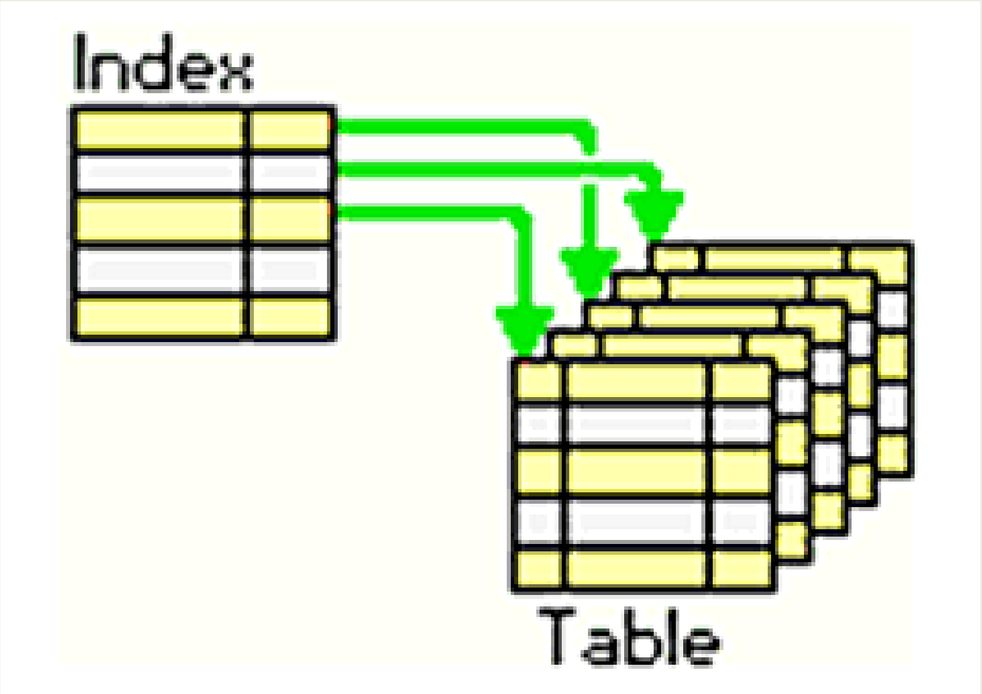
PARTITION

Özetle **verilerin parçalanması** olarak adlandırabileceğimiz bir kavram olan partition, büyük veriye sahip tablolarda I/O, CPU ve Memory tüketimini en az şekilde kullanarak daha fazla işlem yapabilmemize olanak sağlar.

Bu kazanımları sağlarken de mevcut sorgularımızda bir şeyler değiştirmemize gerek kalmadan, ölçeklenebilir ve sürdürülebilir bir mimariye sahip olmamızı sağlar.


List Partitioning
East Sales Region
New York
Virginia
Florida
West Sales Region
California
Oregon
Hawaii
Central Sales Region
Illinois
Texas
Missouri
Range Partitioning
January and February
March and April
May and June
July and August
Hash Partitioning
h1
h2
h3
h4

INDEX KAVRAMI



İndeksleme, veritabanı tablolarındaki verilere daha hızlı erişim sağlamak amacıyla kullanılan bir optimizasyon yöntemidir. Bir veritabanı tablosundaki bir sütun veya sütunlar kombinasyonu üzerinde oluşturulan bir indeks, veriye erişimi hızlandırır. İndeksler, veriyi fiziksel olarak yeniden düzenlemek yerine, bir çeşit **veri yapısı kullanarak erişim hızını artırır**.

İndex olan bir sütunu kullanarak veriye erişim ortalama olarak **O(log(n))** karmaşıklıkta olur.

Ancak her durumda indeksleme kullanmak doğru bir yaklaşım değildir. İndeksler, sorgu performansını artırırken diğer yandan **yazma** işlemlerini yavaşlatabilir. Yeni veri **eklemek** veya varolan veriyi **güncellemek** gibi işlemler, **indekslerin güncellenmesine** neden olur ve bu da performansı olumsuz etkileyebilir.

TAVSİYELER

Okuma işlemlerinin **yoğun** olduğu tüm tablolarda indeksleri oluşturun.

Daha az sayıda arama işlemi, daha fazla sayıda ekleme ve güncelleme işlemi yaptığınız tablolarda gereksiz indekslerden kaçının.

Tam tablo taraması(Full table scan), sorgunun WHERE kısmındaki sütunların kendileriyle ilişkili bir indeks olmadığında gerçekleşir. Bir SQL sorgusunun **WHERE kısmında kullanılan sütunlarda** bir indeks oluşturarak tam tablo taramasını önleyebilirsiniz.

İndeksleme Nasıl Kullanılır?

İndeks oluşturma, veritabanı yönetim sistemine özgü syntax kullanılarak gerçekleştirilir. Temelde, bir `CREATE INDEX` veya `ALTER TABLE` ifadesi kullanarak indeks oluşturabilir veya mevcut bir tabloya indeks ekleyebilirsiniz.

Örneğin:

CREATE INDEX idx_username ON users (username);

veya

ALTER TABLE orders
ADD INDEX idx_customer_orderdate (customer_id, orderdate);

KAYNAKÇA

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

TEŞEKKÜRLER