SQL Clustered index

Clustred Index

Clustred index, veriyi sql'de fiziksel olarak sıraya sokan yapıdır.

Aslında hepimiz clustred index'i tablolarımızda kullanıyoruz.

Tablolarımıza tanımladığımız her bir **Primary ke**y aslında otomatik olarak bir **Clustred index yapısıdır**. Çünkü tablolarımız bu pk'ya göre fiziksel olarak sıralanır.

Clustered index ile ilgili önemli noktalar

- Her tabloda yalnızca 1 adet clustered index olabilir.
- Sql query sonucu sıralı dataları dönerken de clustered indexe göre aynı sırada döner.
- Tablodaki bir clustered index pk olabileceği gibi aynı zamanda birden fazla kolonun birleşiminden oluşan bir yapı da olabilir. Buna composite clustered index denir.

Primary key - Clustered Index farkı nedir?

Primary key

dediğimiz, tablodaki kaydın uniqueliğini garantileyen bir alandır ve kaydın kimliğidir. Tanımlandığı gibi de clustered index özelliği taşır.

Clustered index,

bir veri yapısı, dataya daha hızlı ulaşmak için oluşturulmuş bir indexleme şeklidir.

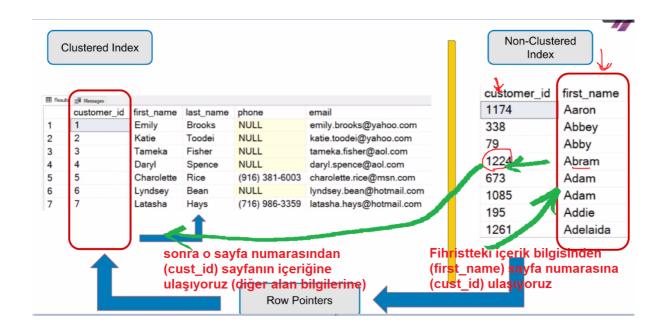
Genel olarak datanın fiziksel sıralamasını düzenleyerek dataya ulaşma süresini optimize etmeyi amaçlarken, keyler ise datanın uniqueliğini sağlar.

Non-Clustered Index

- Bir kolonu Non-clustered index olarak indexlediğinizde, arka tarafta yeni bir tablo oluşur ve bu tablo sizin indexlediğiniz kolona karşılık kolon adresini tutar. Yani bir nevi pointer yapısı gibi düşünebilirsiniz.
- Non-clustered indekste verilere direkt erişilemez. Elde edilen indeksleme yapısına erişmek için kümelenmiş indeks yapısı kullanılmış olur.

 Verileri herhangi bir alana göre sıralandığında erişim kümelenmiş indeks üzerinden anahtar değer referans alınarak yapılır.

Kitapların başında içerik kısmı vardır. Bu içerik kısmında her bir konu başlığının hangi sayfa numarasında veya sayfa numaraları aralığında olduğunu gösterir. Siz kitabı açtığınızda önce içerik sayfasına bakarsınız. **Daha sonra aradığınız içeriğin sayfasını ya da sayfa aralığını öğrenip direkt olarak bu sayfalara geçersiniz**. Non-cluster index de tam olarak bunu yapmakta.



COMPOSITE CLUSTERED INDEX

Tablodaki bir clustered index PK olabileceği gibi aynı zamanda birden fazla kolonun birleşiminden oluşan bir yapı da olabilir. Buna composite clustered index denir.

Aşağıdaki query ile 2 kolona bağlı clustered index yaratmış olduk.

Bu query'nin bize söylediği şey, *tablomdaki kayıtları Gender'a göre diz*, *genderi aynı olanları da artan sırada salary'e göre diz*. Siz insertlerinizi ne şekilde atarsanız atın, dizilim bu yönde olacaktır.

create clustered index myClusteredIndex On tblEmployee (Gender Desc, Salary Asc)

Pyhton DB API

daha önce crs= conn.cursor() ile cursor tanımıştık. o yüzden şimdi tekrar tanımlamadık. cursor a crs.execute() metodu uyguladık. Neyi execute ettirdik? "INSERT TestA (FirstName, LastName) VALUES ('Bob', 'Marley')"

```
crs= conn.cursor()

crs.execute("INSERT TestA (FirstName, LastName) VALUES ('Bob', 'Marley')")
```

#

conn.commit()--> conn.autocommit = True yaptığımız için her seferinde conn.commit() yapmamıza gerek kalmadı.

Aslında her defasında execute metodunun içine aşağıdaki gibi bildiğimiz SQL query si yazıyoruz. o da onu çalıştırıyor. yaptığımız bu ;

```
309
310 DELETE FROM tablo adi
311 WHERE secilen alan adi=alan degeri
312
313 DELETE FROM Personel
314 WHERE Sehir='İstanbul'
315 AND id = 3
316
317
```

ÖNEMLİ:

fetchone() başlangıçta ilk satırı, sonra her çalıştırıldığında bir sonraki satırı getiriyor.

fetchone()

```
cursor.fetchall()
```

bir sorgu sonucunun tüm satırlarını getirir. Tüm satırları bir tuple listesi olarak döndürür. Alınacak kayıt yoksa boş bir liste döndürülür.

```
cursor.fetchmany(size)
```

size değişkeni tarafından belirtilen satır sayısını döndürür. Tekrar tekrar çağrıldığında, bu yöntem bir sorgu sonucunun sonraki satır kümesini getirir ve bir tuple listesi döndürür.

```
cursor.fetchone()
```

tek bir kayıt döndürür. satır yoksa None döndürür.