Sevdenur YILMAZ

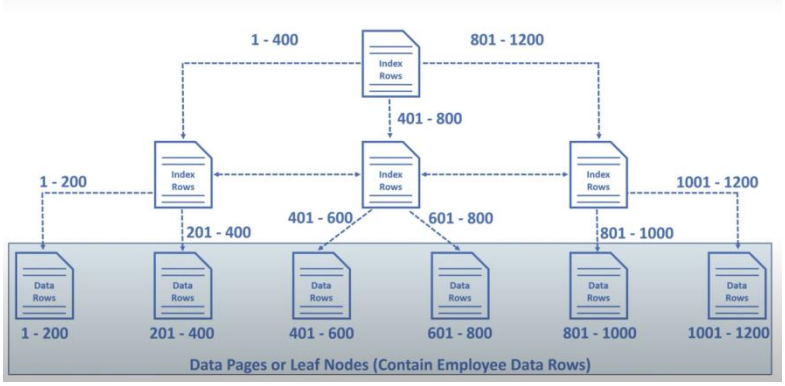
INDEX NEDİR?

Indexleme bir veritabanın tablolarındaki veriler sorgulandığında daha az veri okuyarak çok daha hızlı bir şekilde veriye ulaşmayı amaçlayan ve işlem sonucunu daha hızlı döndüren yapılardır.

NOT: SQL Unique ve SQL Primary Key özellikleri de bir indextir.

Indexler hakkında klasik bir örnek olarak telefon rehberi verilebilir. Telefon rehberindeki kayıtların sıralı olmaması durumunda, yani her kaydın telefon defterinde rastgele tutulması durumunda, arayacağımız bir isim için tüm rehberi gezmemiz gerekecek. Ama rehberinizdeki kayıtlar sıralı olsaydı, aradığımız ismin rehberin ortasındaki isimden ileride mi yoksa geride mi olduğuna bakabilirdik. Bu şekilde aradığımız verileri eleyerek bir kaç adımda istediğimiz sonuca ulaşabilirdik. Bu örnekteki gibi verinin sıralı tutulmasını sağlayan nesnelere index denir.

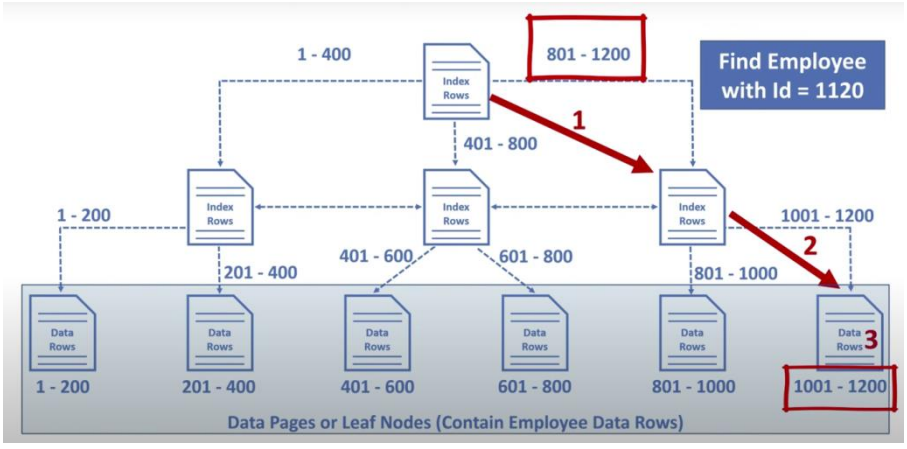
Employee tablosunda 1200 satır var ve her veri sayfasında 200 veri olduğunu varsayalım. Böylece, ilk veri sayfasında 1 ile 100, ikincide 201-400, üçüncüde 401 ile 600 satırımız olur ve bu böyle devam eder.



Ağacın tepesindeki düğüme kök düğüm denir. Kök düğüm ile yaprak düğümler arasındaki düğümlere ara düzeyler denir. Kök ve orta düzey düğümler, dizin satırlarını içerir. Her dizin satırı bir anahtar değeri(elimizdeki tabloda EmployeeID) ve Dengeli Ağaçtaki(B- Tree) bir orta düzey sayfaya veya yaprak bir düğümdeki bir veri satırına işaretçi içerir. Dolayısıyla bu ağaç benzeri yapı sorgu motorunun verileri hızlı bir şekilde bulmasına yardımcı olan bir dizi işaretçiye sahiptir.

Örneğin EmployeeID= 1120 olan çalışan satırını bulmak istediğimizi varsayalım.

- Böylece database engine 1.adımda kök düğümde başlar ve sağdaki dizin düğümü seçer. Çünkü veritabanı engine, 801'den 1200'e kadar EmployeeID içeren bir düğüm olduğunu bilir.
- 2.adımda oradan, en sağda bulunan yaprak düğümü seçer çünkü 1001'den 1200'e kadar Employee veri satırları bu yaprak düğümde bulunur. Yaprak düğümdeki veri satırları EmployeeID'ye göre sıralanır.
- 3. adımda SQL Server aradığımız verileri bulabilir.



Örneğin, bir kitapta index yoksa ve sizden belirli bir bölümü bulmak istersek ilk sayfasından sonuna kadar her sayfasına bakmamız gerekecek. Diğer yandan o kitapta index olduğunu varsayalım. O zaman indexi kullanarak bölümün sayfa numarasını aratıp direkt devam edeceksiniz.

Aslına index yazmanın faydası sorguya yardımcı olacak bir index yoksa sorgunun performansını büyük ölçüde artırabilir. Database engine nin tablodaki her satırı taraması(scan) gerekir. Buna scan table(tablo taraması) denir. Database terminology ve table cans performans için kötüdür. Bu nedenle bir sorgu ne zaman bu sorguya yardımcı olacak bir index olmadığında table scan ile sonuçlanır. Bu durumda database engine veriyi bulmak için baştan sona her satırı taraması gerekir. Eğer tablo boyutu büyükse o zaman sorgunun performansını kötü etkiler.

Örnek olarak oluşturulan Employee tablosunda bu sorguya yardımcı olacak bir index yoksa her satırı benzer şekilde taramam gerekecek.

```
select * from tblEmployee
where Salary between 15000 and 30000
```

91 %

Results | Messages

	ID	Name	Salary	Gender
1	1	Ali	30000	Male
2	2	Göktuğ	15000	Male
3	3	Işıl	25000	Female
4	4	Sevde	28000	Female
5	7	Şeyda	29000	Female

```
insert into tblEmployee values(3, 'Işıl', 25000, 'Female')
insert into tblEmployee values(4, 'Sevde', 28000, 'Female')
insert into tblEmployee values(1, 'Ali', 30000, 'Male')
insert into tblEmployee values(2, 'Göktuğ', 15000, 'Male')
insert into tblEmployee values(6, 'Deniz', 13000, 'Female')
insert into tblEmployee values(7, 'Şeyda', 29000, 'Female')
insert into tblEmployee values(5, 'Taha', 45000, 'Male')
insert into tblEmployee values(8, 'Can', 59000, 'Male')
```

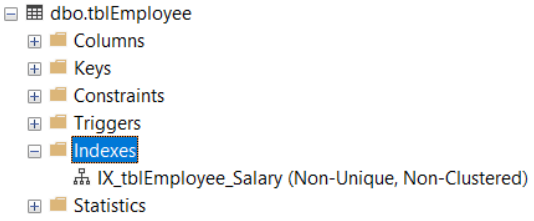
Tabloya değerleri ekledik. ID kolonuna artan veya azalan sırada değer girmediğimiz halde tablomuzda ID lerin sıralanmış şekilde olduğunu görüyoruz.

	ID	Name	Salary	Gender
1	1	Ali	30000	Male
2	2	Göktuğ	15000	Male
3	3	Işıl	25000	Female
4	4	Sevde	28000	Female
5	5	Taha	45000	Male
6	6	Deniz	13000	Female
7	7	Şeyda	29000	Female
8	8	Can	59000	Male

- Bir tablo üzerinde index oluşturmak için ‘CREATE INDEX’ yapısını kullanıyoruz

```
CREATE INDEX IX_tblEmployee_Salary
ON tblEmployee (Salary ASC);
```

- tblEmployee de oluşturduğumuz Salary kolonunda artan sırada bir index oluşturuyoruz.
- Oluşturulan index non-clustered yapıda oldu.



- `sp_Helpindex tblEmployee`

Mevcut olan veya bunu için oluşturulmuş tüm indeksleri bulmak için Helpindex store procedure ünü kullanabiliriz.

- tblEmployee tablosunda oluşturulan tüm indexleri bulmak istiyorum;

	index_name	index_description	index_keys
1	IX_tblEmployee_Salary	nonclustered located on PRIMARY	Salary

- İndeksi silmek istersek index üzerinde bir tablo olduğu için tablo adını belirterek silebiliriz.

```
drop index tblEmployee.IX_tblEmployee_Salary
```

INDEX TYPES

1. Clustered
2. Nonclustered
3. Unique
4. Filtered
5. XML
6. Full Text
7. Spatial
8. Columnstore
9. Index with included columns
10. Index on computed columns

1.CLUSTERED INDEX

Verilerin fiziksel sırasını belirler. Bu nedenle bir tablonun yalnızca clustered indexi olabilir

ID primary key olarak işaretlenmiş bu yüzden otomatik olarak bu sütunda clustered index oluşturacaktır.

```
create table tblEmployee
(
    ID int primary key,
    Name nvarchar(50),
    Salary int,
    Gender nvarchar(10),
    City nvarchar(50)
)
```

- `execute sp_helpindex tbEmployee`

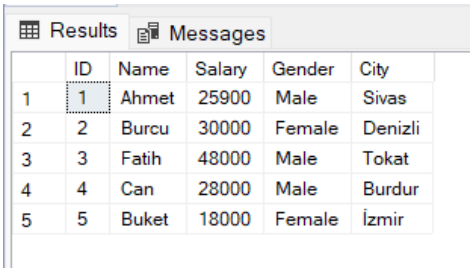
	index_name	index_description	index_keys
1	PK__tbEmploy__3214EC272128265B	clustered, unique, primary key located on PRIMARY	ID

ID sütununda clustered index olduğunu gösteriyor. Bunu biz yapmadık otomatik olarak oluşturdu. Tablonun herhangi bir clustered indexi yoksa o sütunda bir clustered index oluşturun

NOT: Primary key değerleri karışık olarak eklendi. ID değeri 1 olanı değeri eklediğimde bu kaydı üste alıp 3 olan değeri aşağı itiyor. ID değerini 4 eklediğimde 3 değerinden sonra geliyor. En sonra ID'ye göre verileri yeniden düzenliyor.

```
insert into tblEmployee values (3, 'Fatih', 48000, 'Male', 'Tokat')
insert into tblEmployee values (1, 'Ahmet', 25900, 'Male', 'Sivas')
insert into tblEmployee values (4, 'Can', 28000, 'Male', 'Burdur')
insert into tblEmployee values (2, 'Burcu', 30000, 'Female', 'Denizli')
insert into tblEmployee values (5, 'Buket', 18000, 'Female', 'İzmir')
```

Verileri ekledikten sonra; `select * from tbEmployee` kullanarak tablonun içeriğine bakıyoruz. Karışık girilen ID değerlerinin artan şekilde olduğunu görüyoruz.



Results | Messages

	ID	Name	Salary	Gender	City
1	1	Ahmet	25900	Male	Sivas
2	2	Burcu	30000	Female	Denizli
3	3	Fatih	48000	Male	Tokat
4	4	Can	28000	Male	Burdur
5	5	Buket	18000	Female	İzmir

Böylece clustered index, verilerin bu tabloda nasıl saklandığını bildiğinizi belirler.

Clustered index, verilerin soyadına göre düzenlendiği bir telefon rehberine benzer. Bir tablonun yalnızca bir clustered index e sahip olabilir.

Ancak index, bir telefon rehberinin soyadı ve isme göre düzenlenmesi gibi birden çok sütun (composite index) içerebilir.

Telefon rehberi tıpkı bir composite index gibidir. Çünkü temelde numaraların önce soyadına göre düzenlendiğini ve ardından kişi veya kuruluşlar için benzer soyadları varsa o zaman veriler adlarına göre düzenlenir

Employee tablosunda Gender ve Slary için bir clustered index oluşturmak istiyorum.

Bu şu anlama geliyor; verileri önce cinsiyete göre sonra maaşa göre sıralamak istiyorum. Bu tabloda veriler bu sıraya göre düzenlenmeli. Bunu yapmaya çalıştığımızda ID sıralaması bozulacak yani belirli bir zamanda yalnızca belirli bir şekilde düzenleyebilirsiniz, bu nedenle tablo başına yalnızca bir clustered indexe sahip olabilirsiniz.

```
Create Clustered Index IX_tblEmployee_Gender_Salary
on tblEmployee(Gender desc, Salary asc)
```

```
Cannot create more than one clustered index on table 'tblEmployee'. Drop the
existing clustered index 'PK__tblEmplo__3214EC2713729534' before creating another.
```

Bu tabloda clustered index oluşturmak istediğimizde birden fazla clustered index oluşturamayacağımızı belirten bir hata alırız.

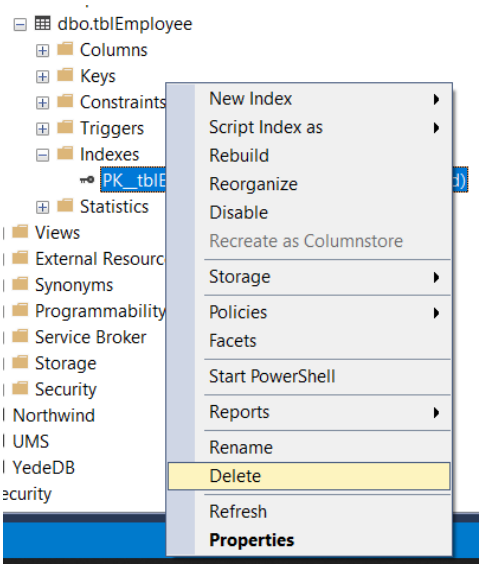
Bir tabloya bir index bırakmak için kullanılan yol;

```
drop index tblEmployee.PK__tblEmplo__3214EC2713729534
```

```
An explicit DROP INDEX is not allowed on index
'tblEmployee.PK__tblEmplo__3214EC2713729534'. It is being used for PRIMARY KEY constraint
enforcement.
```

Bunu yapmak istediğimizde bu indexte açık bir indexe izin verilmiyor hatası alırız. Çünkü primary key kısıtlaması uygulaması için kullanılır. Primary key kısıtlamasını uygulamak için unique clustered index kullanılır.

Indexi güvenli bir şekilde silmek için yapılabilecek şey Object Explorer'da tblEmployee tablosunda Index klasörünü açıp sağ tıklayıp Delete butonu ile silinebilir.



Böylece ID sütununda clustered indexi sildik. Şimdi tekrar Clustered Index oluşturmaya çalışalım.

```
Create Clustered Index IX_tblEmployee_Gender_Salary
on tblEmployee(Gender desc, Salary asc)
```

Şu anda yalnızca bir tane clustered indexe sahibiz. Birden fazla clustered indexe sahip olmak mümkün değildir. Ancak o tek clustered indexin o index anahtarları içinde birden fazla sütuna sahip olması mümkündür. Bu nedenle şimdi oluşturacağım küme indexi şu şekilde iki sütuna sahip olacak. Index anahtarları cinsiyet ve maaş. Cinsiyeti önce azalan, sonra cinsiyet içindeki maaş artan sırada düzenlenecektir.

```
select * from tblEmployee
```

	ID	Name	Salary	Gender	City
1	1	Ahmet	25900	Male	Sivas
2	4	Can	28000	Male	Burdur
3	3	Fatih	48000	Male	Tokat
4	5	Buket	18000	Female	İzmir
5	2	Burcu	30000	Female	Denizli

Önceden ID sütununu artan sıraya göre düzenlenmişti. Ama artık cinsiyet ve maaş sütunlarında clustered index oluşturduğumuz için veriler bu tabloda cinsiyet önce azalan sonra o cinsiyet içindeki maaş artan sırada düzenlenecektir.

Cinsiyet ve maaş sütunlarında clustered index oluşturduğumuz için artık veriler bu tabloda önce cinsiyet azalan sıraya göre düzenleniyor, sonra cinsiyet içinde daha sonra artan düzende düzenleniyor.

2.NON-CLUSTERED INDEX

tblEmployee adlı tablo için Name sütununda non-clustered index oluşturmak istersek,

Nonclustered index oluşturduğunuzda nonclustered indexler ders kitabındaki bir indexe benzer. Yani ders kitabındaki indexe bakarsanız index ayrı bir yerde saklanır ve veriler ayrı yerde saklanır. Örneğin kitabın başında index bölümüne sahibiz, eğer belirli bölüme gitmek istersek önce indexi kontrol edeceksin 5. Bölüm 400. Sayfaya gideceksin böylece index ayrı olarak saklanır ve verinin kendisi ayrı olarak saklanır. Bir telefın rehberi veya sözlükte veriler alfabetik sırayla düzenlenir, temelde verilerin kendisi sizin sahip olmadığınız şekilde düzenlenir.

```
Create NonClustered Index IX_tblEmployee_Name
on tblEmployee(Name)
```

Elimizde olduğu için indexin kendisi burada ayrı depolanıyor. Name sütununda bir nonclustered index oluşturuldu. Name sütunu artan sırada düzenlenir ve sonra bildiğiniz her satır adresin bir adresi vardır.

	ID	Name	Salary	Gender	City
1	1	Ahmet	25900	Male	Sivas
2	4	Can	28000	Male	Burdur
3	3	Fatih	48000	Male	Tokat
4	5	Buket	18000	Female	İzmir
5	2	Burcu	30000	Female	Denizli

Bir tablo birden fazla nonclustered indexe sahip olabilir. Tıpkı bir kitap başındaki bölümlere göre bir indexe sahip olabilir ve belki kitabın sonunda ortak terimlere göre başka bir indexe sahip olabilir. Kaç indexe sahip olabileceğiniz konusunda kısıtlama yoktur.

NOT: Clustered index yalnızca bir indexe sahiptir. Çünkü verileri yalnızca bir şekilde düzenleyebilirsiniz, eğer başka sütuna göre düzenlemeye çalışırsak ilk sütundaki sıralamayı kaybedersiniz bu nedenle bir tabloda yalnızca bir clustered indexe sahip olabilirsiniz

Nonclustered indexte verilerin kendisi, bu index anahtarına göre artan veya azalan düzende düzenlenir. Indexi oluştururken bunu belirtebilirsiniz. Ancak bu nonclustered indexi hiçbir şekilde etkilemeyecektir. Clustered index anahtarı tarafından belirlenen tablonun kendisinde hangi verilerin depolanacağı ve tablo üzerinde clustered index yoksa o zaman verilerin belirli şekilde sıralanmasının garanti edilmediğini bilirsiniz.

CLUSTERED INDEX ve NONCLUSTERED INDEX ARASINDAKİ FARKLAR

- Tabloda yalnızca bir clustered indexe sahipken, birden fazla nonclustered indexe sahip olabilirsiniz
- Clustered index, nonclustered indexe göre daha hızlıdır. Çünkü seçilen sütun dizinde yoksa Clustered indexin tabloya geri başvurması gerekir.
- Clustered index, tablodaki satırların depolama sırasını belirler ve bu nedenle ek disk alanı gerektirmez, ancak Nonclustered index tablodan ayrı olarak depolandığında ek depolama alanı gerekir

3. UNIQUE INDEX

Unique index, dizin anahtarının yinelenen değerler içermemesini ve dolayısıyla tablodaki veya görünümdeki her satırın bir şekilde benzersiz olmasını sağlar.

UNIQUE kısıtlaması oluşturmak ile kısıtlamadan bağımsız benzersiz bir dizin oluşturmak arasında önemli bir fark yoktur. Veri doğrulama aynı şekilde gerçekleşir ve sorgu optimize edici, bir kısıtlama tarafından oluşturulan veya manuel olarak oluşturulan benzersiz bir dizin arasında ayrım yapmaz.

Uniqueness(Benzersizlik), hem clustered hem de nonclustered indexlerin bir özelliği olabilir.

```
CREATE UNIQUE INDEX AK_UnitMeasure_Name
ON Production.UnitMeasure (Name)
```

4. FILTERED INDEX

Bu index türünde ise tüm tabloya index tanımlamak yerine, belirlenen koşula uyan veriye index tanımlanır. Hem performansı arttırır hem de index bakım maliyeti düşük olur. Normal bir non-clustered index'e göre daha az yer kaplar.

```
CREATE NONCLUSTERED INDEX IX_IndexName ON IndexName (Column1, Column2) WHERE …
```

5. XML INDEX

XML dizinleri, xml veri türü sütunlarında oluşturulabilir. Sütundaki XML örnekleri üzerinden tüm etiketleri, değerleri ve yolları indeksler ve sorgu performansından faydalanırlar.

6. SPATIAL INDEX

Bir spatial index, geometri veri türünün bir sütunundaki spatial objects (spatial data) üzerinde belirli işlemleri daha verimli bir şekilde gerçekleştirme yeteneği sağlar. Spatial index, nispeten maliyetli spatial işlemlerin uygulanması gereken nesnelerin sayısını azaltır.

SQL Server'da Spatial Index B-Tree kullanılarak oluşturulur bu, dizinlerin 2-boyutlu Spatial Index B-tree 'nin doğrusal sırasında temsil etmesi gerektiği anlamına gelir.

7. FULL-TEXT INDEX

Sql Server'da metin tabanlı aramalarda, like operatörünün performansı özellikle metin boyutu büyüdükçe düşmektedir. Bunun dışında index kısıtlamalarında bahsettiğimiz gibi 900 byte sınırı da index oluşturmamızı kısıtlar. Özellikle büyük boyutlu veri içeren alanlarda (varchar(max), nvarchar(max), xml, text) hızlı arama yapabilmek için bu index türünü kullanabiliriz. Aslında bu Sql Server'ın sunduğu bir servistir.