SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Bartul Brajković, 0036507098

30. ožujka 2021.

LABORATORIJ PROFILA 2

Odjeljak Sustavi baza podataka

3. Vježba

2. zadatak

2.1. zadatak

Izvršavanjem sljedećeg upita:

```
SELECT object_name(object_id) AS table_name, page_count, record_count
FROM  sys.dm_db_index_physical_stats  (DB_ID('labprof3'),  OBJECT_ID('salesOrderItem'),  0,
NULL, 'DETAILED');
```

Dobiju se statistički podaci za tablicu salesOrderItem. Podaci koji se dobiju su broj blokova relacije salesOrderItem te broj n-torki te relacije.

U predavanjima iz predmeta SBP vrijednosti page_count i record_count označeni su simbolima B(r), odnosno N(r).

2.2. zadatak

Nakon kreiranja indeksa nad atributom orderQty nad tablicom salesOrderItem, izvršio sam sljedeću naredbu:

```
SELECT name AS index_name, index_type_desc, index_depth, index_level, page_count,
record_count
FROM sys.dm_db_index_physical_stats (
                                      DB_ID('labprof3')
                                    , OBJECT_ID('salesOrderItem')
                                    , NULL
                                    , NULL
                                    , 'DETAILED') AS t1
JOIN sys.indexes AS t2
 ON t1.index_id = t2.index_id
        AND t1.object_id = t2.object_id
        AND name IS NOT NULL;
```

Rezultat naredbe je sljedeći:

Results | Messages

	index_name	index_type_desc	index_depth	index_level	page_count	record_count
1	i1	NONCLUSTERED INDEX	2	0	271	121317
2	i1	NONCLUSTERED INDEX	2	1	1	271

Statistički podaci koje sam dobio izvršavanjem naredbe su: tip stvorenog indeksa, dubina B-stabla za stvoreni indeks, razina indeksa, broj blokova relacije salesOrderItem te broj n-torki te relacije.

U predavanjima je indeks_depth označen simbolom d(idx).

Za jedan indeks dobio sam više redaka zato što se za indeks vraća po jedan redak za svaku razinu B-stabla u svakoj particiji. Statistike za svaku razinu B-stabla su odvojene, a budući da B-stablo ima dvije razine dobio sam dva retka u rezultatu. Također razina indeksa se broji od dna prema vrhu.

3. zadatak

3.1. zadatak

```
CREATE STATISTICS stat1 ON salesOrderItem (orderQty) WITH FULLSCAN;
```

Nakon provjere stvorenih statističkih podataka dobio sam sljedeći rezultat:

Results | Messages

	name	auto_created	user_created	name	name
1	stat1	0	1	salesOrderItem	orderQty

Navedena naredba stvara statistiku nad atributom orderQty tablice salesOrderItem imena stat1.

WITH FULLSCAN označava da će SQL Server pri stvaranju statistike skenirati 100% vrijednosti stupca orderQty.

3.2. Zadatak

```
UPDATE STATISTICS salesOrderItem WITH ALL;
```

Navedena naredba ažurira statistiku kako bi se osiguralo da se queryji kompajliraju s najnovijom statistikom (ali ažuriranje statistika uzrokuje da se queryji rekompajliraju pa to ne bi trebalo raditi prečesto). WITH ALL označava da će se ažurirati sve kreirane statistike.

3.3. Zadatak

```
DBCC SHOW_STATISTICS (salesOrderItem, stat1) WITH STAT_HEADER;
```

Results | Messages

	Name	Updated	Rows	Rows Sampled	Steps	Density	Average key length	String Index	Filter Expression	Unfiltered Rows	Persisted Sample Percent
1	stat1	Mar 31 2021 12:56AM	121317	121317	40	0.5	4	NO	NULL	121317	0

Name = ime statistike kreirane nad tablicom salesOrderItem.

Rows = ukupan broj redaka u tablici od zadnjeg puta kada je statistika ažurirana.

Rows sampled = ukupan broj redaka koji se uzimaju za kalkulaciju statistike.

Steps = broj koraka u histogramu

3.4. Zadatak

```
DBCC SHOW_STATISTICS (salesOrderItem, stat1) WITH DENSITY_VECTOR;
```

Results | Messages

	All density	Average Length	Columns
1	0.02439024	4	orderQty

Density = 1/distinct values

All density = prikazuje gustoću za svaki prefiks stupca u statističkom objektu

Colums = ime stupca za prefiks za koji će biti prikazani podaci All density i Average Length

3.5. Zadatak

```
DBCC SHOW_STATISTICS (salesOrderItem, stat1) WITH HISTOGRAM;
```

Results | Messages

	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
1	1	0	74954	0	1
2	2	0	14200	0	1
3	3	0	10058	0	1
4	4	0	7494	0	1
5	5	0	4386	0	1
6	6	0	3298	0	1
7	7	0	1735	0	1
8	8	0	1521	0	1
9	9	0	837	0	1
10	10	0	768	0	1

RANGE_HI_KEY = gornja granica vrijednosti stupca za korak histograma

RANGE_ROWS = procijenjeni broj redaka čija vrijednost u stupcu spada u korak histograma (bez gornje granice)

EQ_ROWS = procijenjeni broj redaka čija je stupčana vrijednost jednaka gornjoj granici koraka histograma.

DISTINCT_RANGE_ROWS = procijenjeni broj redaka s različitim vrijednostima stupca unutar koraka histograma

AVG_RANGE_ROWS = prosječan broj redaka s kopijama (duplikatima) vrijednosti stupca unutar histogramskog koraka.

3.6. zadatak

```
DROP STATISTICS salesOrderItem.stat1;
```

Navedena naredba uništi objekt sa statističkim podacima

3.7. zadatak

Nakon kreiranja indeksa nad atributom orderQty tablice salesOrderItem imena i1, stvorio se statistički objekt imena i1 s istim podacima kao i statistički objekt kreiran u 3.1. s razlikom da sada ovaj objekt nije user created.

```
CREATE INDEX i1 ON salesOrderItem (orderQty);
```

Results | Messages

	name	auto_created	user_created	name	name
1	i1	0	0	salesOrderItem	orderQty

3.8. zadatak

Uništavanjem indeksa uništi se i objekt sa statističkim podacima koji je nastao u trenutku stvaranja indeksa.

```
DROP INDEX i1 ON salesOrderItem;
```

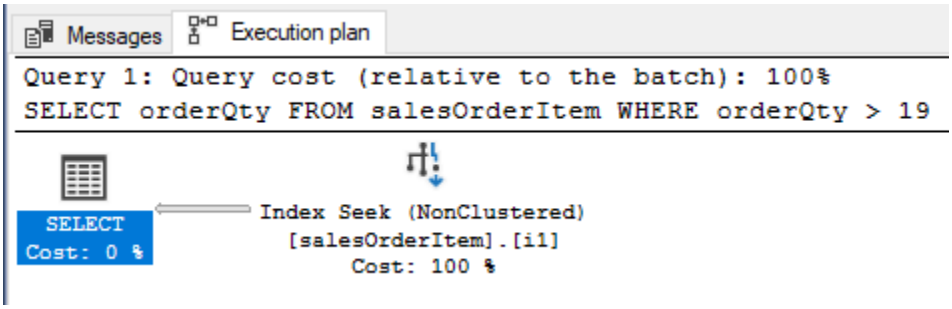
5. zadatak

5.2. zadatak

5.2. Napisati upit koji koristi SQL Server operator index-seek bez RID lookup (u predavanjima se naziva key-only index scan).

```
CREATE INDEX i1 ON salesOrderItem (orderQty)
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT orderQty FROM salesOrderItem WHERE orderQty > 19;
DROP INDEX i1 ON salesOrderItem
```

Plan izvršavanja:



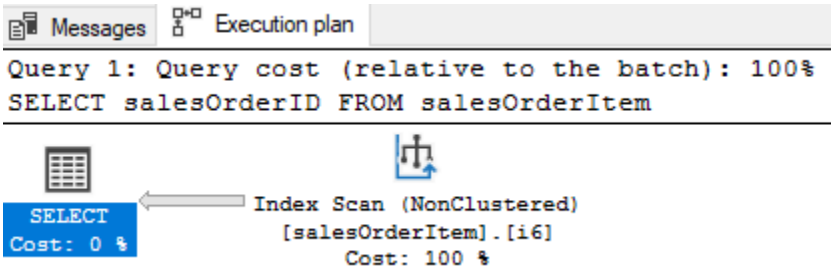
Za predikat selekcije postoji odgovarajući indeks. Pomoću B-stabla pronalaze se (Indeks Seek) odgovarajuće kazaljke na n-torke u podacima s blokovima. Vrijednosti atributa koje treba pročitati odnose se samo na atribute koji se nalaze u listovima B-stabla. To znači da SUBP neće trebati dohvaćati n-torke ili dijelove n-torki iz blokova s podacima. Ključevi u B-stablu sadrže sve podatke koji su navedeni u SELECT listi. (key-only index scan)

5.3. zadatak

5.3. Napisati upit koji koristi SQL Server operator index-scan (u predavanjima se naziva key-only index scan).

```
CREATE INDEX i6 ON salesOrderItem (salesOrderID)
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT salesOrderID FROM salesOrderItem;
DROP INDEX i6 ON salesOrderItem
```

Plan izvršavanja:



Napravili smo indeks nad atributom salesOrderID. NonClustered Index Scan se događa kada je nad svim atributima u SELECT listi napravljen indeks, ali je selektivnost upita preslaba da bi imali Indeks Seek. Čitaju se sve n-torke relacije jer nema uvjeta u upitu.

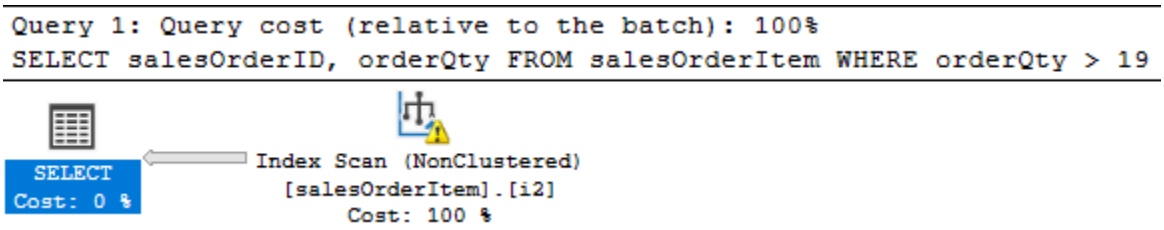
5.4. zadatak

5.4. Proučiti što su to indeksi s dodanim atributima (Indexes with Included Columns). Indeks koji se koristio u rješenju zadatka 5.1, u ovom zadatku treba prepraviti u indeks s dodanim atributima, na način koji će osigurati izvršavanje istog upita korištenjem samo operatora index-seek (dakle bez dodatnog RID Lookup). Usporedite veličinu memorije koju koriste indeksi te ukupni procijenjeni trošak izvršavanja upita u zadacima 5.1. i 5.4

```
CREATE INDEX i2 ON salesOrderItem (salesOrderID, orderQty);
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT salesOrderID, orderQty FROM salesOrderItem WHERE orderQty > 19;

DROP INDEX i2 ON salesOrderItem;
```

Plan izvršavanja:



Ukoliko napravimo indeks koji će uključivati stupce koji su navedeni u SELECT listi optimizator upita će koristiti jedino non-clustered indeks kako bi vratio tražene podatke. Takav indeks će značajno povećati performanse upita zato što su svi atributi u upitu uključeni u indeks.

Indeks iz 5.4 zauzima 2680KB memorije, dok indeks iz 5.1 zauzima manje memorije odnosno 2192KB.

Trošak izvršavanja upita je manji u 5.1 nego u 5.4. jer u 5.1. imamo Indeks Seek dok u 5.4. je Indeks Scan.

Table Name	# Records	Reserved (KB)	Data (KB)	Indexes (KB)	Unused (KB)
dbo.pointsInArea1	100,000	2,696	2,680	8	8
dbo.pointsInArea2	120,000	3,272	3,216	8	48
dbo.pointsInArea3	105	72	8	8	56
dbo.salesOrder	31,465	5,896	5,856	8	32
dbo.salesOrderItem	121,317	8,464	5,648	2,680	136

Table Name	# Records	Reserved (KB)	Data (KB)	Indexes (KB)	Unused (KB)
dbo.pointsInArea1	100,000	2,696	2,680	8	8
dbo.pointsInArea2	120,000	3,272	3,216	8	48
dbo.pointsInArea3	105	72	8	8	56
dbo.salesOrder	31,465	5,896	5,856	8	32
dbo.salesOrderItem	121,317	7,952	5,648	2,192	112

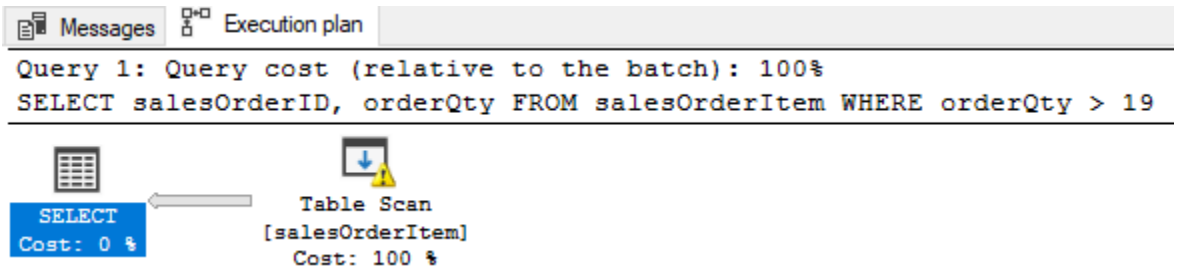
5.5 zadatak

5.5. Napisati upit koji koristi operator table-scan.

Table-scan = čitanje n-torki slijednim čitanjem blokova podataka

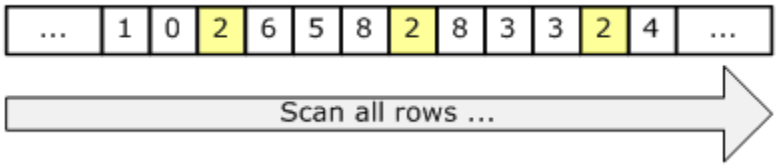
```
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT salesOrderID, orderQty FROM salesOrderItem WHERE orderQty > 19;
```

Plan izvršavanja:

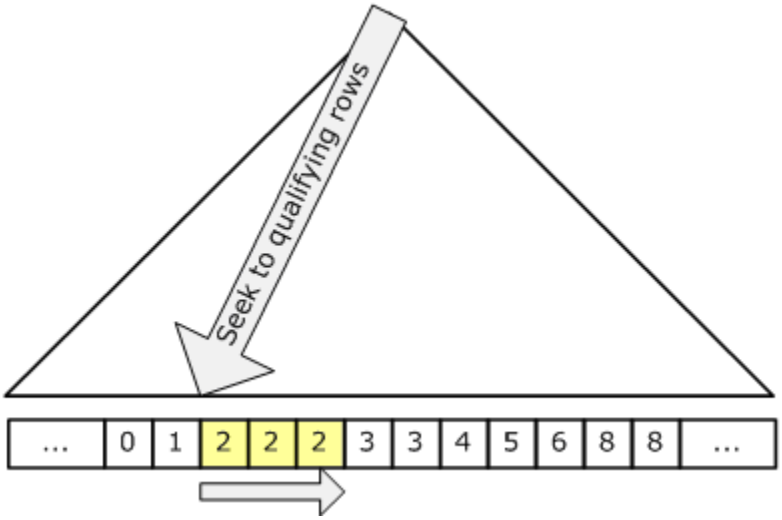


Ukoliko ne kreiramo indeks nad atributom tablice nad kojom ćemo izvršavati upit, u Execution planu vidimo da će se koristiti table scan. To znači da će se prilikom izvršavanja upita skenirati svi redci tablice unutar atributa zadanih u SELECT listi.

Table scan izgleda ovako:



Dok indeks seek izgleda ovako:



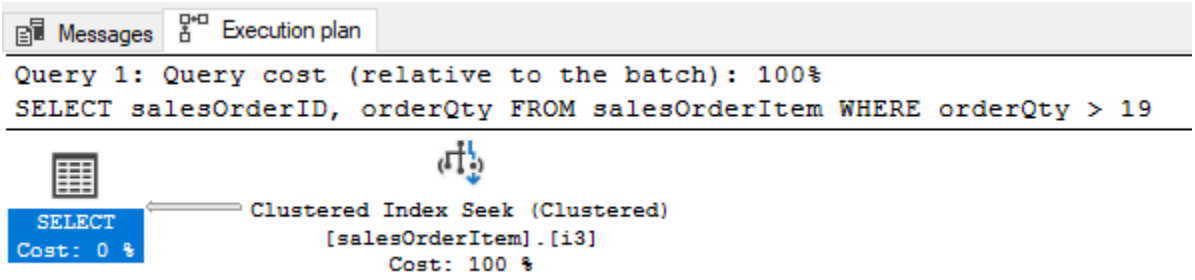
5.6. zadatak

5.6. Napisati upit koji koristi SQL Server operator index-seek (clustered)

```
CREATE CLUSTERED INDEX i3 ON salesOrderItem (orderQty);
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT salesOrderID, orderQty FROM salesOrderItem WHERE orderQty > 19;

DROP INDEX i3 ON salesOrderItem;
```

Plan izvršavanja:



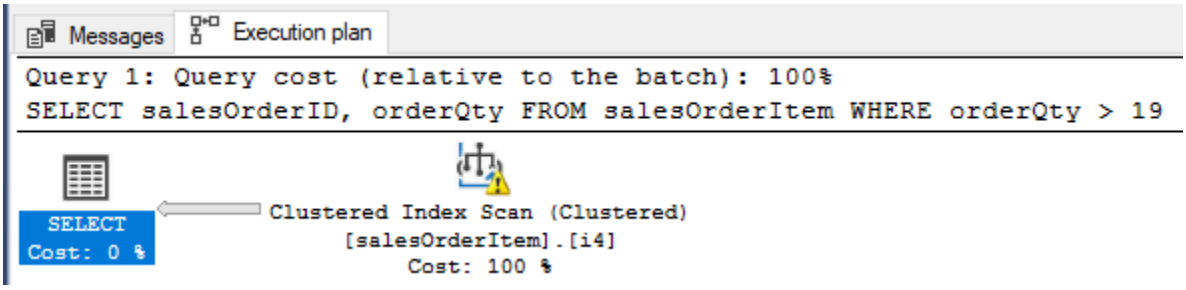
Budući da smo napravili clustered indeks redci su fizički pohranjeni na disk u istom poretku kao indeks. Physical Operation kod ovakvog plana izvršavanja je Clustered index seek.

5.7. zadatak

5.7. Napisati upit koji koristi SQL Server operator index-scan (clustered)

```
CREATE CLUSTERED INDEX i4 ON salesOrderItem (salesOrderID);
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT salesOrderID, orderQty FROM salesOrderItem WHERE orderQty > 19;

DROP INDEX i4 ON salesOrderItem;
```



Napravili smo indeks nad atributom salesOrderID, a pretražujemo podatke (WHERE dio) nad atributom nad kojim nije napravljen indeks. Sql Server će čitati podatke od početka do kraja, odnosno proći će sve redove atributa u WHERE dijelu jer nad tim atributom nije napravljen indeks. Zato je očekivani plan izvršavanja Clustered Indeks Scan, odnosno to je Physical Operation ovog upita.

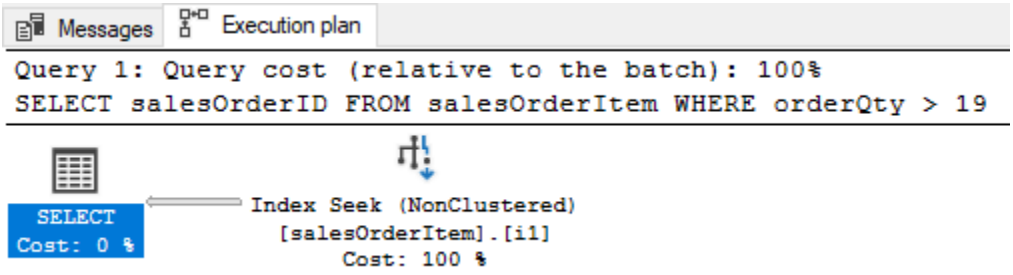
5.8. zadatak

5.8. Napisati upit koji non-clustered indeks koristi za operator index-seek za dohvat n-torki koje se nalaze u listovima drugog, clustered indeksa.

```
CREATE INDEX i1 ON salesOrderItem (orderQty)
CREATE CLUSTERED INDEX i5 ON salesOrderItem (salesOrderID);
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT salesOrderID FROM salesOrderItem WHERE orderQty > 19;
DROP INDEX i1 ON salesOrderItem;

DROP INDEX i5 ON salesOrderItem;
```

Plan izvršavanja:



U ovom zadatku koristimo non-clustered indeks i1 nad atributom salesOrderItem kako bi dohvatili n-torke atributa salesOrderID nad kojim je napravljen clustered indeks.

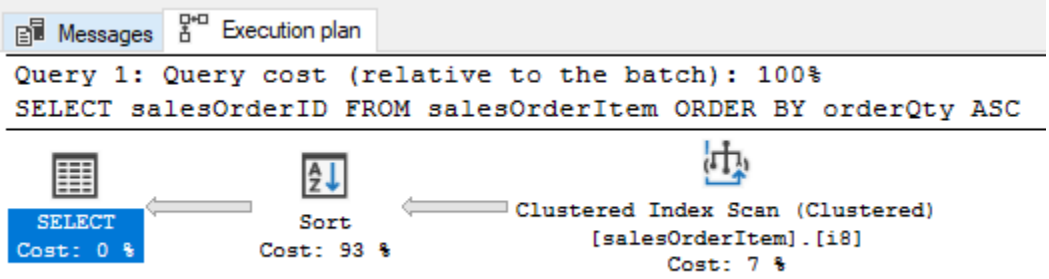
5.9. zadatak

5.9. Napisati upit koji sortiranje dohvaćenih podataka obavlja koristeći SQL Server operator index-scan (u predavanjima se naziva key-only index scan).

```
CREATE CLUSTERED INDEX i8 ON salesOrderItem (salesOrderID)
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT salesOrderID FROM salesOrderItem ORDER BY orderQty ASC;

DROP INDEX i8 ON salesOrderItem
```

Plan izvršavanja:



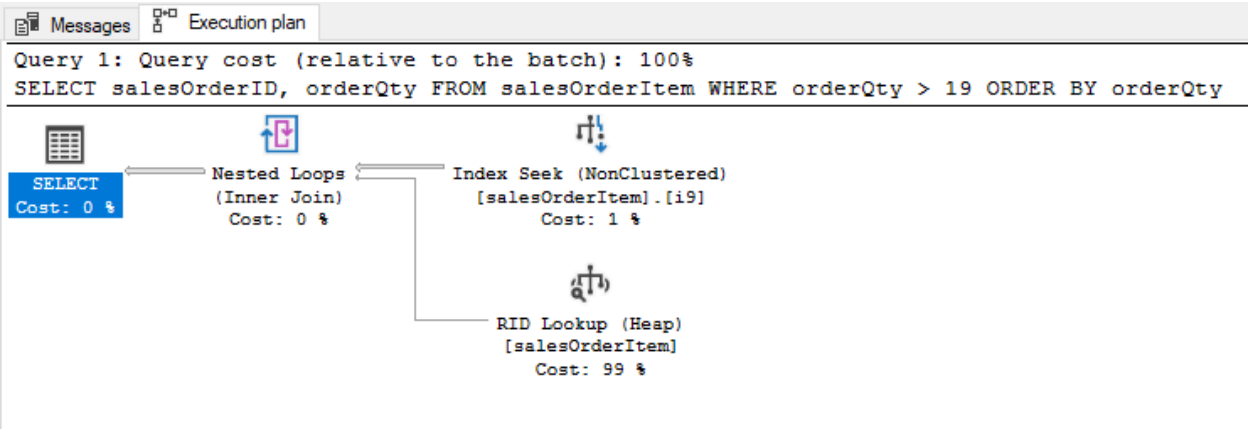
Budući da postoji odgovarajući indeks (B stablo) dobio sam index scan jer ključ indeksa sadrži sve atribute navedene u SELECT listi.

5.10. zadatak

5.10. Napisati upit koji koristi SQL Server operator index-seek + RID lookup, pri čemu se isti indeks koristi i za sortiranje.

```
CREATE INDEX i9 ON salesOrderItem (orderQty);
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT salesOrderID, orderQty FROM salesOrderItem WHERE orderQty > 19 ORDER BY orderQty;
DROP INDEX i9 ON salesOrderItem;
```

Plan izvršavanja:



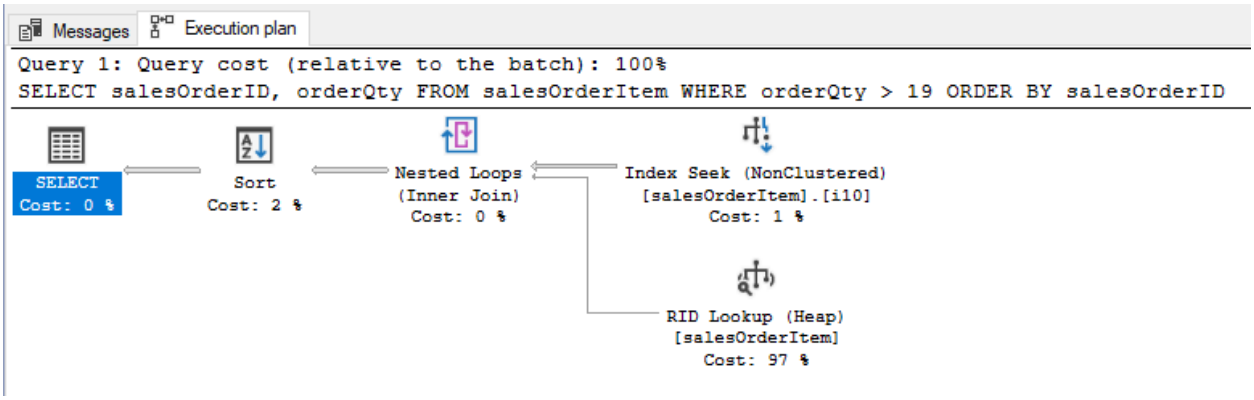
Za predikat selekcije postoji odgovarajući indeks. Pomoću B-stabla pronalaze se (Index Seek) odgovarajuće kazaljke na n-torke u podacima s blokovima, koje se moraju dohvatiti (RID Lookup) iz blokova s podacima jer ključevi u B-stablu ne sadrže sve podatke koji su navedeni u SELECT listi te se isti indeks koristi i za sortiranje.

5.11. zadatak

5.11. Napisati upit koji koristi SQL Server operator index-seek + RID lookup, pri čemu se isti indeks neće moći koristiti i za sortiranje.

```
CREATE INDEX i10 ON salesOrderItem (orderQty);
UPDATE STATISTICS salesOrderItem WITH ALL;
SELECT salesOrderID, orderQty FROM salesOrderItem WHERE orderQty > 19 ORDER BY
salesOrderID;
DROP INDEX i10 ON salesOrderItem;
```

Plan izvršavanja:



Za predikat selekcije postoji odgovarajući indeks. Pomoću B-stabla pronalaze se (Index Seek) odgovarajuće kazaljke na n-torke u podacima s blokovima, koje se moraju dohvatiti (RID Lookup) iz blokova s podacima jer ključevi u B-stablu ne sadrže sve podatke koji su navedeni u SELECT listi, ali se stvoreni indeks ne koristi za sortiranje.