SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Bartul Brajković, 0036507098

30. ožujka 2021.

LABORATORIJ PROFILA 2

Odjeljak Sustavi baza podataka

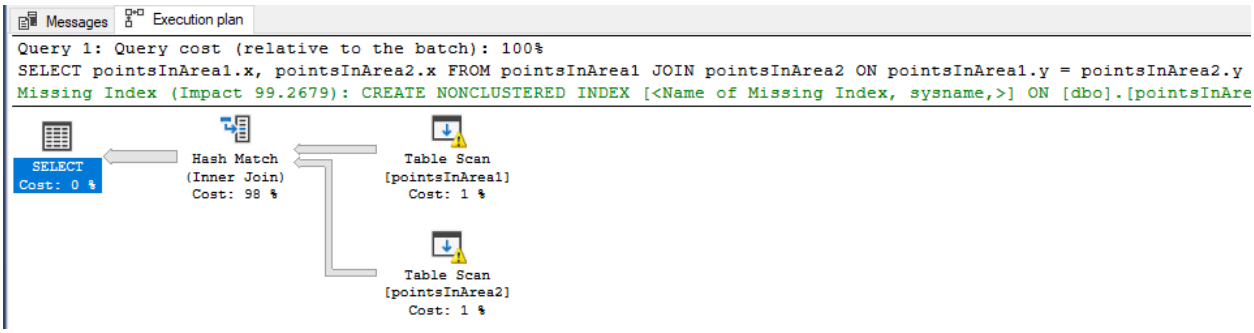
4.

1.1. Zadatak

1.1. Napisati upit koji za spajanje relacija koristi operator hash-match (u predavanjima se naziva hash-join).

```
SELECT pointsInArea1.x, pointsInArea2.x FROM pointsInArea1 JOIN pointsInArea2 ON pointsInArea1.y = pointsInArea2.y;
```

Plan izvršavanja:



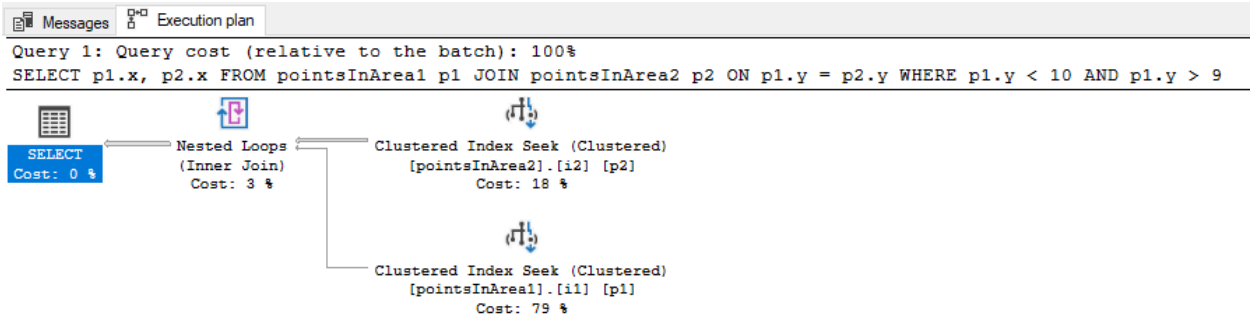
Optimizator je odabrao Hash Match zato što smo u upitu napisali upit spajanja uz uvjet s izjednačavanjem (dio `pointsInArea1.y = pointsInArea2.y`). Kod Hash Match-a funkcija raspršenja h preslikava B-vrijednost n-torke u adresu pretinca (bucket).

1.2. Zadatak

1.2. Napisati upit koji za spajanje relacija koristi operator koji se u predavanjima naziva indexed nested-loop

```
CREATE CLUSTERED INDEX i1 ON pointsInArea1 (y)
CREATE CLUSTERED INDEX i2 ON pointsInArea2 (y)
UPDATE STATISTICS pointsInArea2 WITH ALL;
SELECT p1.x, p2.x FROM pointsInArea1 p1 JOIN pointsInArea2 p2 ON p1.y = p2.y WHERE p1.y < 10 AND p1.y > 9;
DROP INDEX i1 ON pointsInArea1
DROP INDEX i2 ON pointsInArea2
```

Plan izvršavanja:



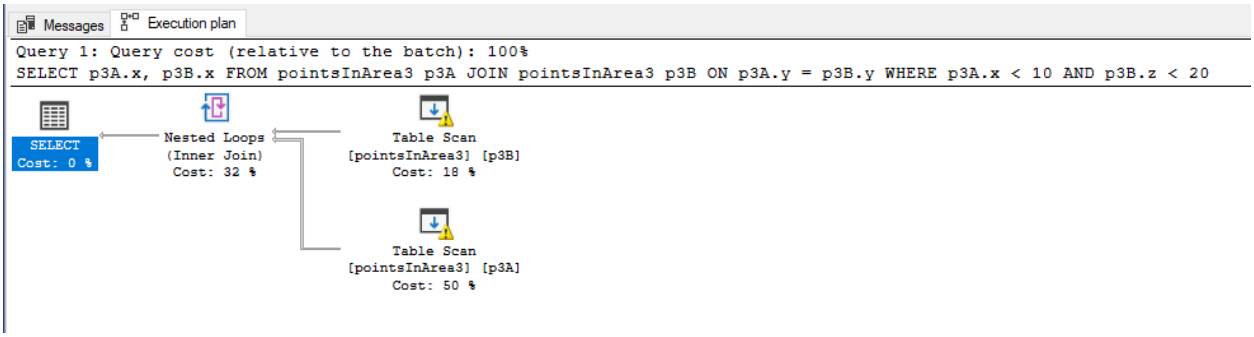
Prvo sam stvorio dva clustered indeksa nad relacijama pointsInArea1 i pointsInArea2 tako da sam indeks stavio na atribute nad kojima ću izvršiti spajanje te ću ih koristiti u WHERE dijelu upita. Optimizator je odabrao Nested Loop zato što sam se pobrinuo da u WHERE dijelu upita stavim dovoljno uvjeta kako bi u rezultatu dobio što manji broj n-torki.

1.3. Zadatak

1.3. Napisati upit koji za spajanje relacija koristi operator nested-loop (bez indeksa).

```
SELECT p3A.x, p3B.x FROM pointsInArea3 p3A JOIN pointsInArea3 p3B ON p3A.y = p3B.y WHERE
p3A.x < 2 AND p3B.z < 5;
```

Plan izvršavanja:



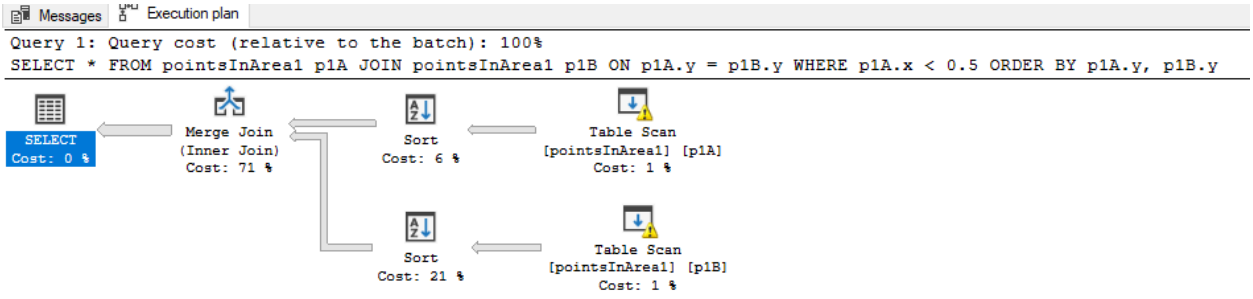
Optimizator je odabrao Nested Loop zato što sam spajanje radio nad tablicom pointsInArea3 koja ima samo 105 n-torki (obje relacije stanu u glavnu memoriju) sam u WHERE dijelu dodao još uvjeta kako bi broj n-torki bio još manji.

1.4. Zadatak

1.4. Napisati upit koji za spajanje relacija koristi operator merge-join. Pomoć: optimizator uporno odabire operator hash-match umjesto merge-join zato jer jednu od relacija koje treba spojiti može u cijelosti pohraniti u primarnu memoriju

```
SELECT * FROM pointsInArea1 p1A JOIN pointsInArea1 p1B ON p1A.y = p1B.y WHERE p1A.x < 0.5
ORDER BY p1A.y, p1B.y;
```

Plan izvršenja:



Možemo uočiti da su obje relacije sortirane po y atributu. Kada nad tim atributom radimo JOIN nad tablicama s velikim brojem n-torki kako se nijedna od relacija u cijelosti ne bi mogla pohraniti u primarnu memoriju te ako nakon toga dodamo uvjet u WHERE dijelu tada optimizator odabire merge-join.

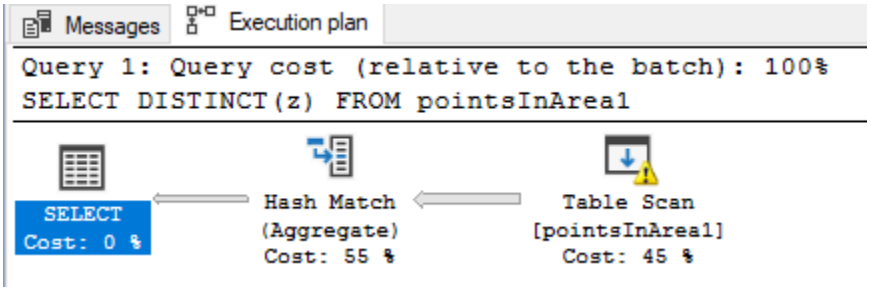
(s desne strane od JOIN postavimo manju relaciju koja bi potencijalno mogla cijela stati u primarnu memoriju)

1.5. Zadatak

1.5. Napisati upit koji eliminaciju duplikata obavlja pomoću raspršenog adresiranja.

```
SELECT DISTINCT(z) FROM pointsInArea1;
```

Plan izvršenja:



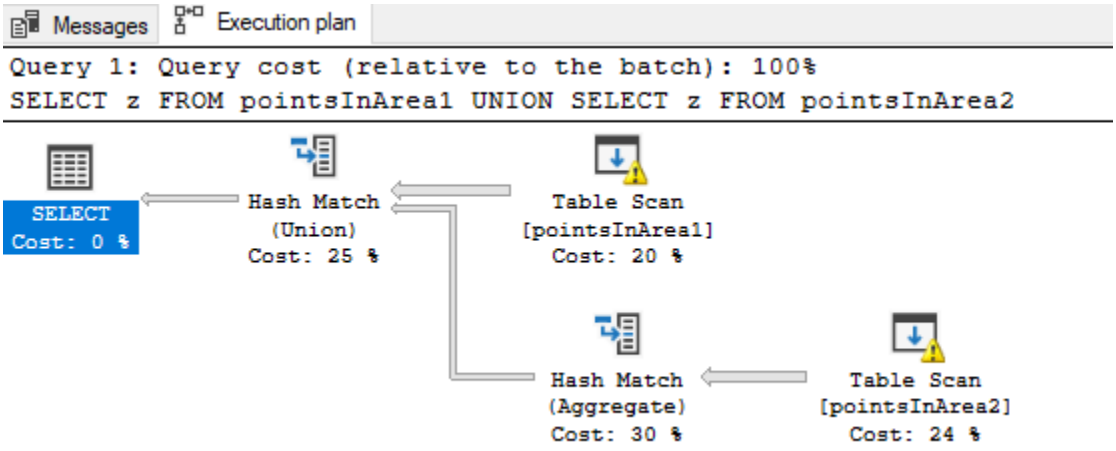
Primjenom funkcije raspršenja DISTINCT(z) nastanu particije r1 , r2 , ..., rM u sekundarnoj memoriji te se za svaku particiju formira se nova in-memory hash table.

1.6. Zadatak

1.6. Napisati upit koji algebarsku operaciju unije obavlja pomoću raspršenog adresiranja.

```
SELECT z FROM pointsInArea1 UNION SELECT z FROM pointsInArea2;
```

Plan izvršavanja:



Primjenom funkcije raspršenja h(t) nastanu particije r1 , r2 , ..., rM, s1 , s2 , ..., sM.

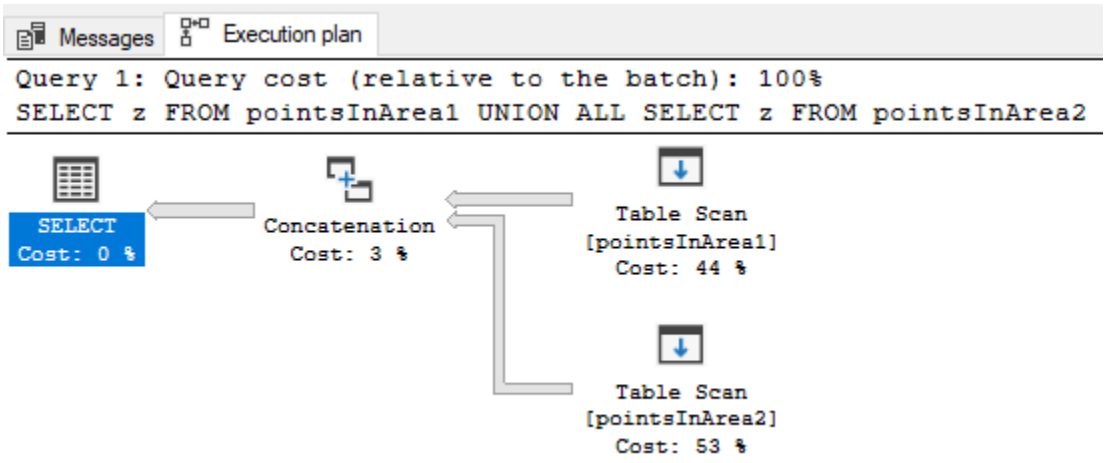
Svaku particiju ri stavimo u in memory hash table te za svaku n-torku particija s gledamo nalazi li se već u in memory hash tableu, ako ne onda ju dodamo. Sadržaj iz in memory hash tablea se doda u konačan rezultat.

1.7. Zadatak

1.7. Napisati primjer upita koji obavlja bag verziju algebarske operacije unije. Koji se fizički operator ovdje koristi i zašto se razlikuje od operatora koji se koristio u zadatku 1.6?

```
SELECT z FROM pointsInArea1 UNION ALL SELECT z FROM pointsInArea2;
```

Plan izvršavanja:



Ovdje se koristi fizički operator konkatenacije zato što UNION ALL, za razliku od običnog UNION-a, ne briše duplikate.