2η ΕΡΓΑΣΙΑ ΣΧΕΔΙΑΣΜΟΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Όνομα:Ανδρέας Πολυχρονάκης
ΑΜ :3170140

Ζήτημα Πρώτο :

1)
Παρατηρούμε ότι πριν την δημιουργία του index είχα :

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(288 rows affected)
Table 'bibrecs'. Scan count 1, logical reads 540, physical reads 3, read-ahead reads 536, lob logical reads 0,

(1 row affected)

Completion time: 2020-04-28T20:23:20.4406038+03:00
```

Στην συνέχεια δημιούργησα το index: Create Index indexbibrecs on [bibrecs]([title]).
Η λογική είναι ότι αντί να πάω να ψάξω τον τίτλο σε ολόκληρο τον πίνακα να κάνουμε δηλαδή tablescan η οποία είναι μια χρονοβόρα διαδικασία μπορώ να αναζητήσω τον τίτλο στο index που ειναι πιο γρήγορη η αναζήτηση,δεδομένου ότι το index είναι μικρότερο από τον πίνακα

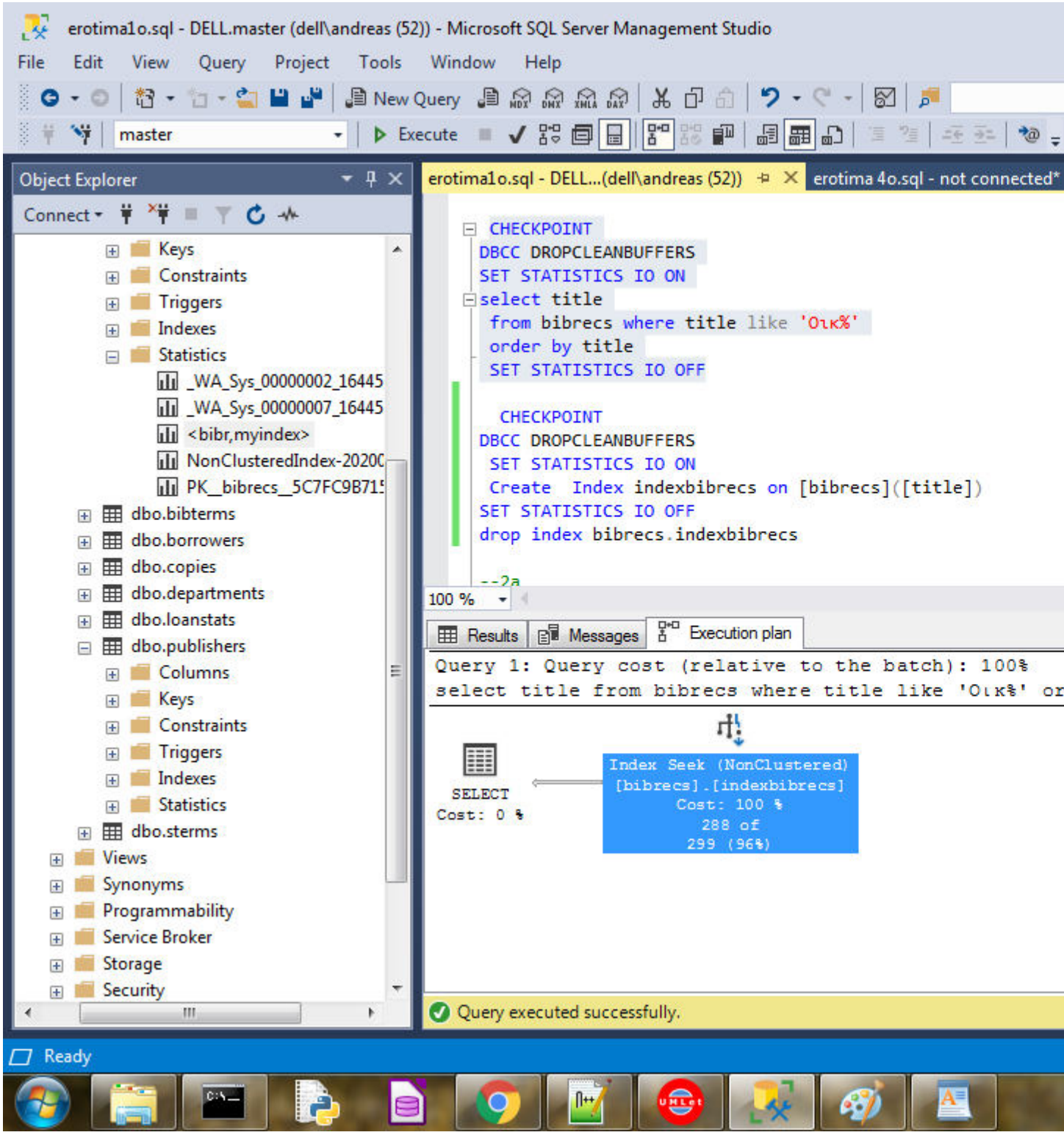
Πραγματί παρατηρώ ότι μέτα την δημιουργία του index έχω:

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(288 rows affected)
Table 'bibrecs'. Scan count 1, logical reads 5, physical reads 2, read-ahead reads 2, lob logical reads 0, lob physical reads 0, lob read-ahead rea

(1 row affected)

Completion time: 2020-04-22T16:48:38.7735776+03:00
```

Όπου φαίνεται οτι τα διαβάσματα έχουν πέσει πάρα πολύ.

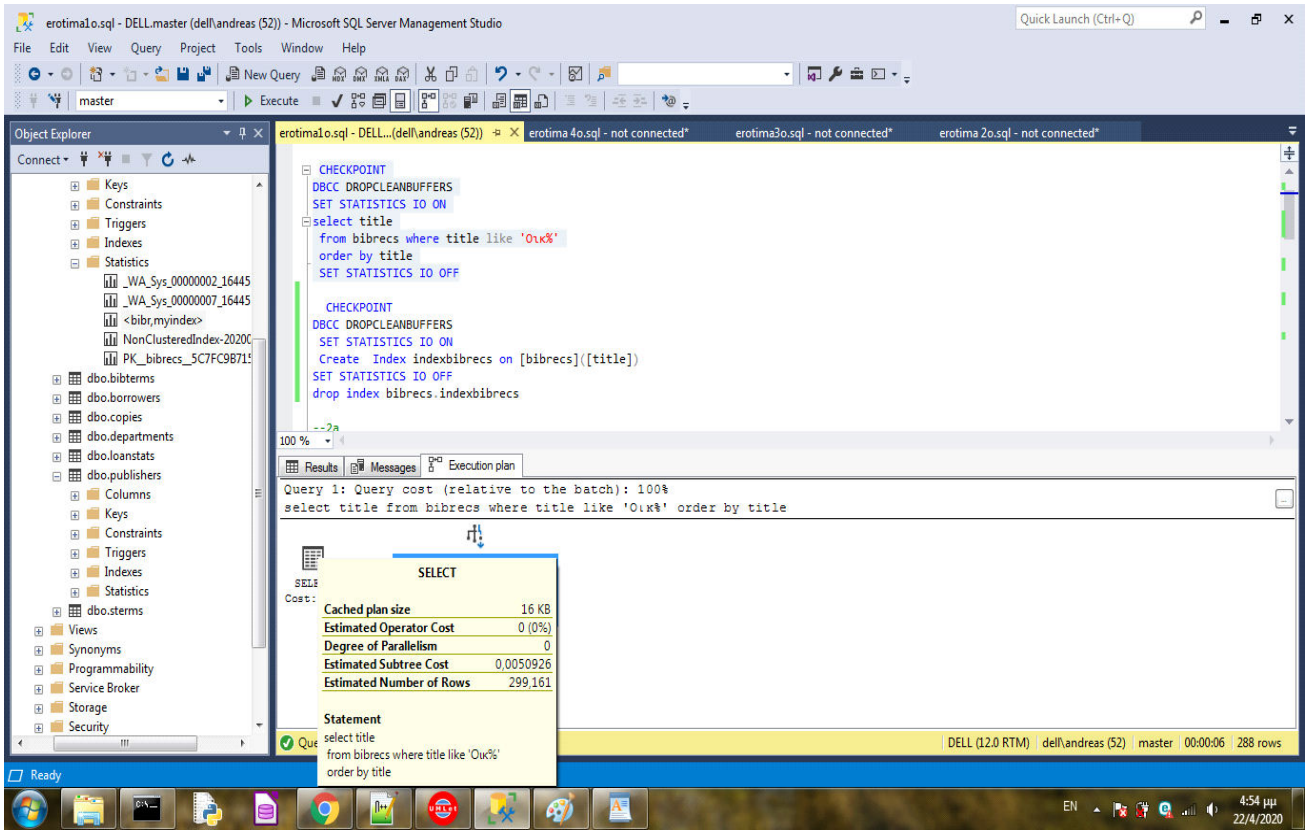
Άν τώρα πάμε να μελετήσουμε και το execution plan

παρατηρούμε ότι αντί να αναζητήσουμε όλους τους τίτλους στον πίνακα μπορούμε με ένα απλό index seek να βγεί πολυ πιο γρήγορα το αποτέλεσμα.



Ένας άλλος τρόπος που καταλαβαίνουμε ότι τα reads έχουν

πέσει είναι και από το estimated subtree cost στην select



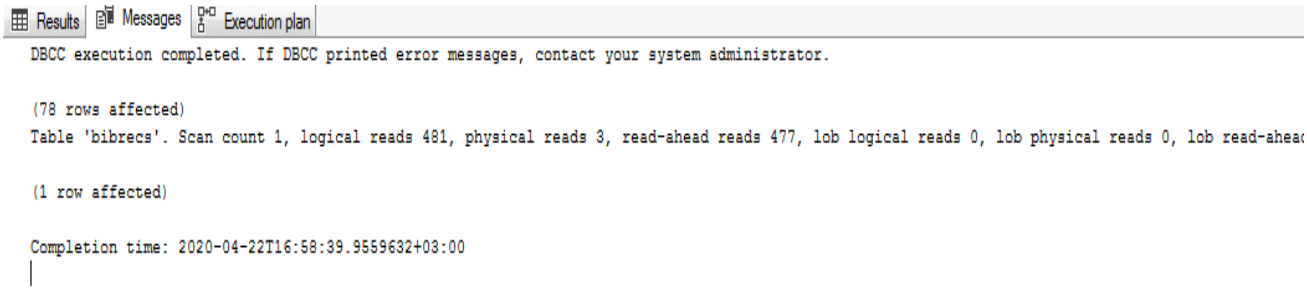
Παρατηρούμε ότι ο αριθμός είναι πολύ μικρός άλλο ένα στοιχείο που μας επιβεβαιώνει ότι το index βοηθάει πάρα πολύ στην βελτίωση εκτέλεσης του query.

2α)

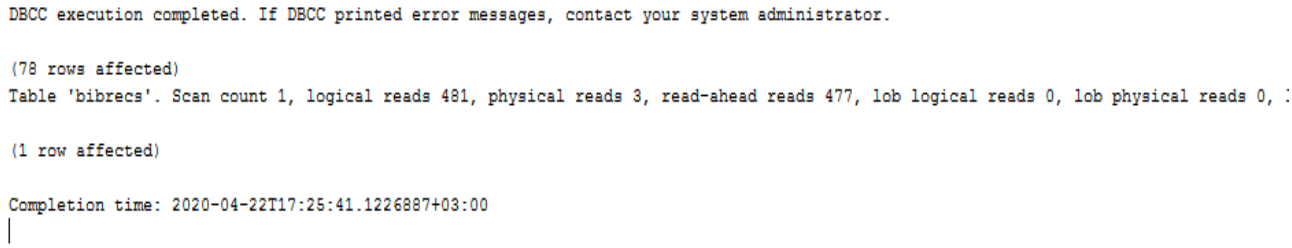
ΚΩΔΙΚΑΣ:

```
Select title
from bibrecs
where title like '%πληροφορική%'
```

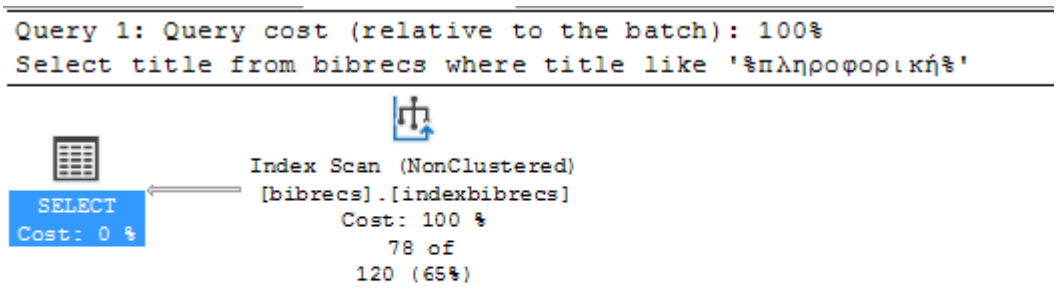
Πριν την δημιουργία του index είχα:



Χρησιμοποιώντας τώρα το index που δημιουργήθηκε προηγουμένως έχω :



Ας εξετάσουμε και το αντίστοιχο execution plan:



Παρατηρούμε ότι γίνεται ένα indexscan στο index που έχει δημιουργηθεί.Στον sql server το index scan στην ουσία είναι το tablescan,οπότε παρατηρώ ότι ενώ δημιούργησα index εντέλη αυτό δεν με βοήθησε στο τελικό μου αποτέλεσμα.

2β)

ΚΩΔΙΚΑΣ

```
Select title,material
from bibrecs
where title='Economics'
```

Παρατηρώ ότι πριν χρησιμοποιήσω query είχαμε:

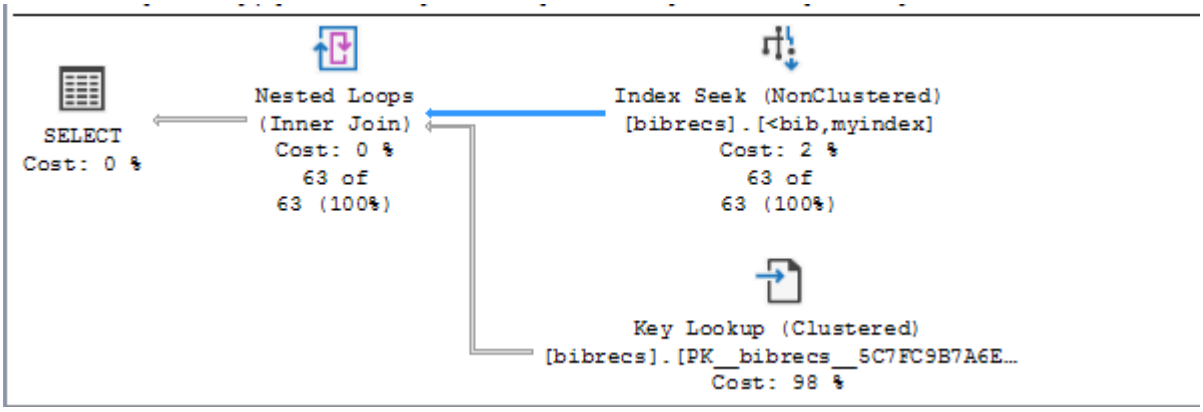
```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
Table 'bibrecs'. Scan count 1, logical reads 481, physical reads 3, read-ahead reads 477, lob logical reads 0, lob physical reads 0,

Completion time: 2020-04-22T17:40:33.4677279+03:00
```

Αν χρησιμοποιούσαμε το index των προηγούμενων ερωτημάτων τότε έχουμε :

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(63 rows affected)
Table 'bibrecs'. Scan count 1, logical reads 367, physical reads 4, read-ahead reads 49, lob logical reads 0, lob physical reads 0,

(1 row affected)

Completion time: 2020-04-22T17:37:01.7786200+03:00
```

Οπότε φαίνεται ότι το index έκανε πιο γρήγορο το query μου.

Ας εξετάσουμε όμως και το execution plan θα δούμε ότι:



Παρατηρώ ότι:Στην αρχή γίνεται index seek το οποιό είναι λόγικό.Δεδομένου ότι έχει δημιουργηθεί ένα index στην στήλη τίτλος τότε στη εντολή(where title='Economics') θα γίνει ένα index seek για να βρω όλους τους τίτλους Economics.Ωστόσο παρατηρώ ότι στην select υπάρχει η εντολή
Select title,material οπότε για κάθε συνθήκη που ισχυεί στο index πρέπει να σκαναριστεί όλος ο πίνακας για να βρεθεί η αντίστοιχη τιμη στην material.Αυτό φαίνεται ξεκάθαρα και από το execution plan.Επειδή το index περιέχει μόνο την στήλη title τότε το πεδίο material θα πρέπει να βρεθεί από τον πίνακα.Αυτή η διαδικασία ονομάζεται key lookup η οποία όμως είναι πολύ χρονοβόρα.

Οπότε μια εναλλακτική λύση θα ήταν να δημιουργηθεί ενα index με γνωρίσματα title,material έτσι ώστε ολόκληρη η διαδικασία να γίνεται στο index όπου είναι πιο μικρό από τον πίνακα και η αναζήτηση πιο γρήγορη.

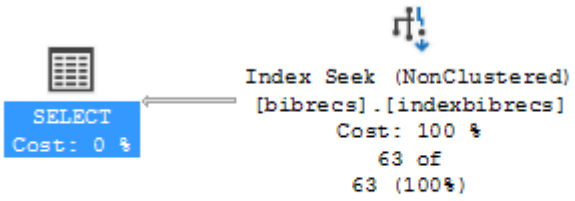
Πράγματι με την δημιουργία του index:

```
Create  Index indexbibrecs on [bibrecs]([title],[material])
```

έχω:

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(63 rows affected)
Table 'bibrecs'. Scan count 1, logical reads 3, physical reads 3, read-ahead reads 0, lob logical reads 0, lob physical reads 0,

(1 row affected)

Completion time: 2020-04-22T18:01:43.1353487+03:00
```

Αν εξετάσουμε και το execution plan βλέπω ότι:



Παρατηρούμε ότι γίνεται απλά ένα index seek.
Έτσι το query μου γίνεται ακόμα πιο γρήγορο από ότι ήταν πριν.

2γ)

ΚΩΔΙΚΑΣ

```
Select title,material
from bibrecs
where title like 'Economics%'
```

Πριν το index είχαμε:

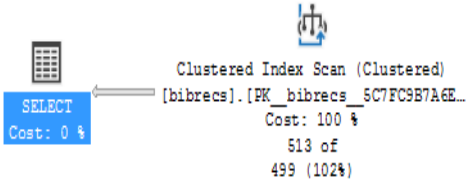
```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(513 rows affected)
Table 'bibrecs'. Scan count 1, logical reads 983, physical reads 3, read-ahead reads 979, lob logical reads 0, lob physical reads 0,

(1 row affected)

Completion time: 2020-04-22T18:18:26.2707248+03:00
```

Μετά την δημιουργία του index:

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(513 rows affected)
Table 'bibrecs'. Scan count 1, logical reads 983, physical reads 3, read-ahead reads 979, lob logical reads 0,

(1 row affected)

Completion time: 2020-04-22T18:22:22.6412444+03:00
```

Άρα παρατηρούμε ότι το index δεν με βοηθάει καθόλου στην προκειμένη περίπτωση.Ο λόγος είναι ότι στην select υπάρχει το material οπότε θα πρέπει να γίνεται key lookup που είναι πολύ χρονοβόρο.Δηλαδή,για κάθε τίτλο που ισχυεί η συνθήκη θα γίνεται ένα tablescan για να βρεθεί η αντίστοιχη τιμή του material.

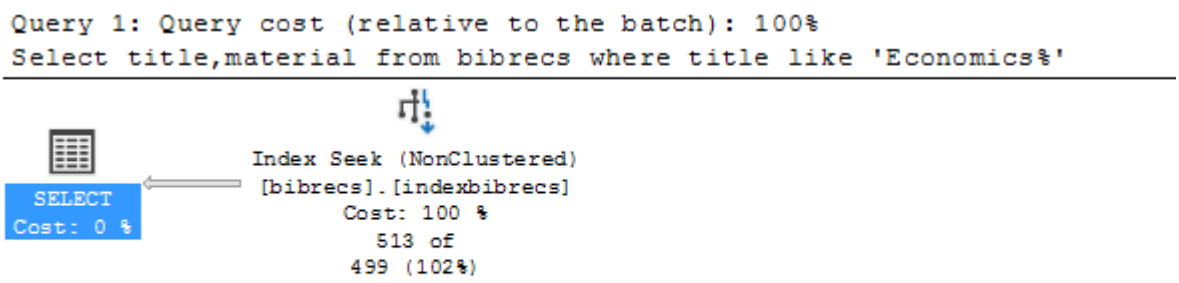
```
Missing Index (Impact 63.6804): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[bibrecs] ([title])
```



Παρατηρούμε ότι πολύ καλύτερη απόδοση θα είχαμε αν συμπεριλαμβάναμε όπως και πριν το index με τα δύο γνωρίσματα

title,material.
Αν χρησιμοποιήσουμε το index που δημιουργήθηκε στο προηγούμενο ερώτημα τότε όπως παρατηρούμε και από τις εικόνες έχουμε μεγάλη βελτίωση στον χρόνο εκτέλεσης.

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(513 rows affected)
Table 'bibrecs'. Scan count 1, logical reads 7, physical reads 2, read-ahead reads 4, lob logical reads 0, lob physical reads 0,

(1 row affected)

Completion time: 2020-04-22T18:28:35.1935532+03:00
```



ΣΗΜΕΙΩΣΗ:Αναλυτικότερη ανάλυση του key lookup:Το key lookup πραγματοποιήται όταν ο query optimizer πραγματοποιέι index seek πάνω σε έναν πίνακα,ωστόσο το index αυτό δεν περιλαμβάνει όλες τις απαραίτητες στήλες που χρειάζονται το αποτέλεσμα(Όπως είδαμε και στο Select title,material).Έτσι ο SQL SERVER αναγκάζεται να ξαναγυρίσει στον clustered index χρησιμοποιώντας το primary key και να σκανάρει τις υπόλοιπες στήλες που χρειάζονται για να ολοκληρωθεί το αποτέλεσμα.Η διαδικασία αυτη ονομάζεται key lookup και αποτελεί μια χρονοβόρα διαδικασία.

Ζήτημα 2:

2α)

ΚΩΔΙΚΑΣ

```
select title,lang
from bibrecs,publishers
where bibrecs.pubid=publishers.pubid AND
pubname='Κλειδάριθμος'
```

Πριν το index είχαμε:

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.
Table 'bibrecs'. Scan count 1, logical reads 536, physical reads 3, read-ahead reads 532, lob logical reads 0,
Table 'publishers'. Scan count 1, logical reads 23, physical reads 1, read-ahead reads 21, lob logical reads 0,

Completion time: 2020-04-22T18:42:47.2262867+03:00
```

Έτσι προκειμένου να βελτιωθεί όσο γίνεται περισσότερο το query δημιουργήθηκαν δύο indexes:

Για τον πίνακα publishers:

```
create index[pub]
on [publishers] ([pubname],[pubid])
```

Για τον πίνακα bibrecs:

```
CREATE INDEX [bib]
ON [bibrecs] ([pubid],[title],[lang])
```

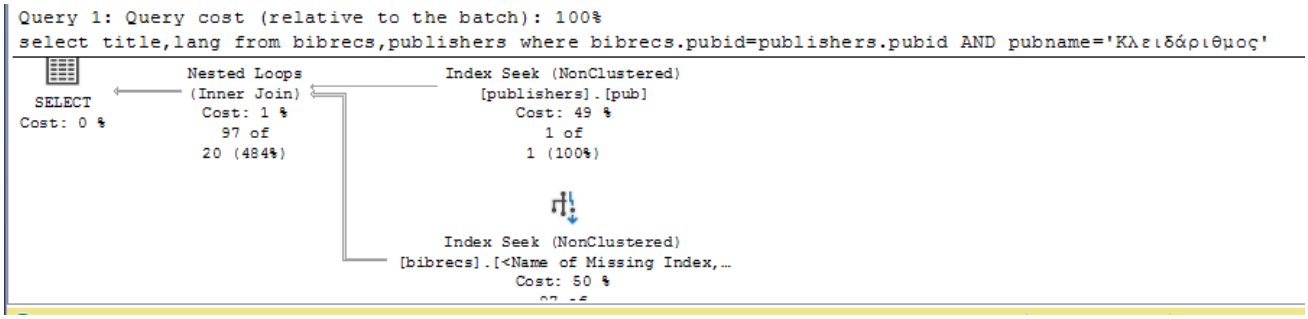
Έτσι έπειτα από την δημιουργία των δύο indexes έχω:

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(97 rows affected)
Table 'bibrecs'. Scan count 1, logical reads 4, physical reads 4, read-ahead reads 0, lob logical reads 0,
Table 'publishers'. Scan count 1, logical reads 2, physical reads 2, read-ahead reads 0, lob logical reads

(1 row affected)

Completion time: 2020-04-22T18:58:00.4345192+03:00
```

Παρατηρώ ότι τα δύο αυτά indexes έχουν βελτιώσει κατά πολύ την εκτέλεση του query μου.

Ο λόγος είναι ότι πριν δημιουργηθούν τα indexes γινόταν tablescan προκειμένου να βρεθούν οι τιμές που ικανοποιούν την συνθήκη,το οποιό όπως καταλαβαίνουμε είναι μια χρονοβόρα διδαδικασία.Μετά όμως από την δημιουργία των indexes γίνονται δύο index seeks.Ένα για τον πίνακα bibrecs και ένα για τον πίνακα publishers το οποίο είναι μία πιο γρήγορη διαδικασία.

Ας εξετάσουμε και το execution plan:



Παρατηρούμε ότι γίνονται δύο index seeks.

Ο λόγος που δημιούργησα το index του bibrecs ως εξής:

```
CREATE INDEX [bib]
ON [bibrecs] ([pubid],[title],[lang])
```

ήταν διότι αν δημιουργούσα το index μόνο με τήν τιμή που υπήρχε στην συνθήκη where(pubid) τότε επειδή η select περιέχει title και lang που ανήκουν στον πίνακα bibrecs για κάθε ένα index seek που θα έκανα στην στήλη pubid θα έπρεπε να γίνεται key lookup στον πίνακα bibrecs το οποίο όπως αναφέραμε είναι μία χρονοβόρα διαδικασία .
Ώστοσο με την δημιουργία του παραπάνω index γλυτώνουμε όλον αυτον τον κόπο και γίνεται απλώς ένα index seek.

2β)

ΚΩΔΙΚΑΣ

```
Select depname,count(lid) AS totallid
from departments,loanstats,borrowers
where borrowers.bid=loanstats.bid AND
```

```
borrowers.depcode=departments.depcode and loandate like
'2000%'
Group by depname
```

Πριν την δημιουργία του index είχαμε:

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(10 rows affected)
Table 'departments'. Scan count 0, logical reads 20, physical reads 1, read-ahead reads 0, lob logical reads 0, lob physical reads
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0,
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, l
Table 'borrowers'. Scan count 1, logical reads 71, physical reads 1, read-ahead reads 69, lob logical reads 0, lob physical reads 0
Table 'loanstats'. Scan count 1, logical reads 480, physical reads 1, read-ahead reads 478, lob logical reads 0, lob physical reads

(1 row affected)

Completion time: 2020-04-22T19:30:16.2002386+03:00
```

Έπειτα προκειμένου να βελτιωθεί η απόδοση δημιούργησα τα εξής indexes.

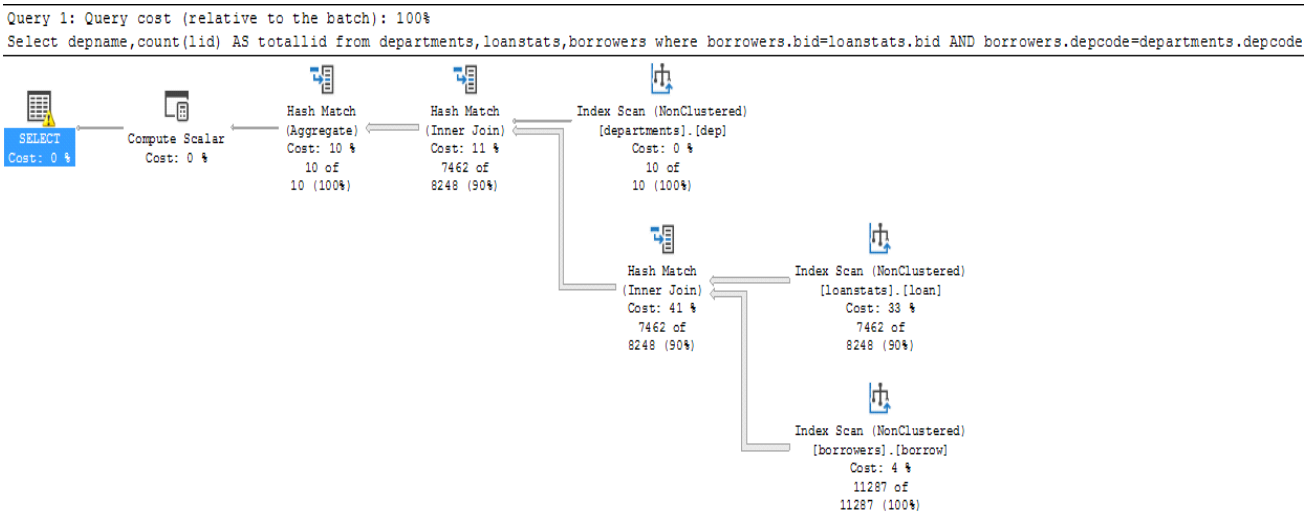
```
1)Create Index borrow on [borrowers]([bid],[depcode])
```

```
2)create index[loan]
on [loanstats] ([lid],[loandate],[bid])
```

Δημιουργώντας τα indexes παρατηρώ ότι έχει βελτιωθεί ο χρόνος εκτέλεσης του query μου το οποίο φαίνεται από την παρακάτω εικόνα:

```
(10 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, l
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lo
Table 'borrowers'. Scan count 1, logical reads 22, physical reads 1, read-ahead reads 20, lob logical reads 0,
Table 'loanstats'. Scan count 1, logical reads 195, physical reads 1, read-ahead reads 193, lob logical reads
Table 'departments'. Scan count 1, logical reads 2, physical reads 1, read-ahead reads 0, lob logical reads 0,

(1 row affected)
```

Ας εξετάσουμε και το αντίστοιχο execution plan:



Έτσι επιβεβαιώνεται ότι με την δημιουργία των indexes έχει βελτιωθεί η εκτέλεση του query.
Αυτό γίνεται διότι όπως και προηγουμένως δεν χρειάζεται να γίνει tablescan η οποία είναι μια χρονοβόρα διαδικασία.

2γ)

```
Select title,lang,author
from sterms,bibauthors,authors,bibterms,bibrecs
where( bibrecs.bibno = bibterms.bibno and
bibterms.tid=sterms.tid and
bibrecs.bibno=bibauthors.bibno and
bibauthors.aid=authors.aid and sterms.term='Databases')
```

Πριν την δημιουργία indexes έχω:

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(1263 rows affected)
Table 'authors'. Scan count 0, logical reads 3852, physical reads 31, read-ahead reads 164, lob logical reads (
Table 'bibauthors'. Scan count 779, logical reads 3523, physical reads 4, read-ahead reads 204, lob logical rea
Table 'bibrecs'. Scan count 0, logical reads 3768, physical reads 1, read-ahead reads 367, lob logical reads 0,
Table 'bibterms'. Scan count 1, logical reads 4, physical reads 4, read-ahead reads 0, lob logical reads 0, lob
Table 'sterms'. Scan count 1, logical reads 2, physical reads 2, read-ahead reads 0, lob logical reads 0, lob p

(1 row affected)

Completion time: 2020-04-22T20:03:04.0267919+03:00
```

Έπειτα προκειμένου να μειωθεί όσο γίνεται περισσότερο ο

χρόνος εκτέλεσης του query μου δημιουργήθηκαν τα παρακάτω indexes:

1)
```
CREATE INDEX [stermindex]
ON [sterms] ([term])
```

2)
```
Create index author on bibauthors (bibno,aid)
```

3)
```
CREATE INDEX [myindex3]
ON [bibrecs] ([bibno]) INCLUDE ([title],[lang])
```

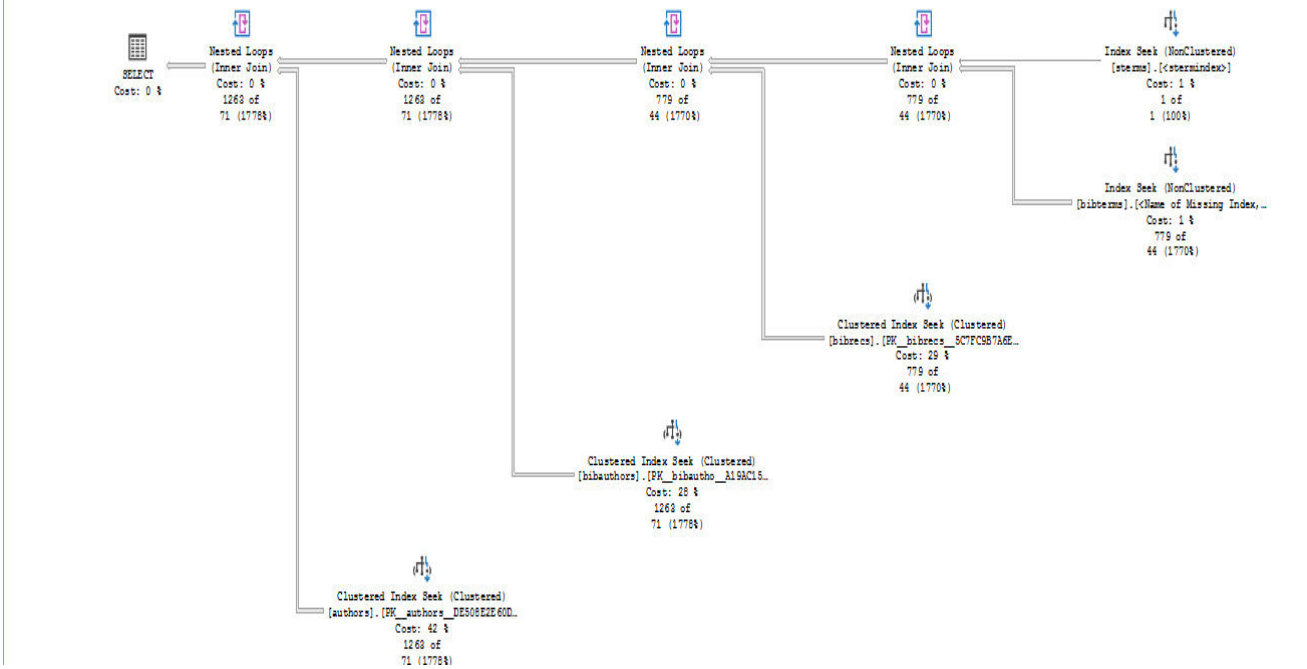
ΣΗΜΕΙΩΣΗ: Με την εντολή `INCLUDE ([title],[lang])` δημιουργείται ένα Index στον πίνακα bibrecs πάνω στην στήλη του bibno ωστόσο επείδη για το τελικό μου αποτέλεσμα χρειάζομαι και τις στήλες title,lang με τον όρο include τις περιλαμβάνω στο index που έχει δημιουργηθεί.
Αυτό έχει σαν αποτέλεσμα να μην χρειαστεί να γίνει για κάθε bibno ένα key lookup το οποίο θα κοστίσει πολύ.

Έτσι μετά τα δημιουργημένα indexes έχω:

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(1263 rows affected)
Table 'authors'. Scan count 0, logical reads 4041, physical reads 8, read-ahead reads 187, lob logical reads 0,
Table 'bibauthors'. Scan count 779, logical reads 3324, physical reads 2, read-ahead reads 122, lob logical reac
Table 'bibrecs'. Scan count 0, logical reads 2768, physical reads 1, read-ahead reads 275, lob logical reads 0,
Table 'bibterms'. Scan count 1, logical reads 4, physical reads 4, read-ahead reads 0, lob logical reads 0, lob
Table 'sterms'. Scan count 1, logical reads 2, physical reads 2, read-ahead reads 0, lob logical reads 0, lob ph

(1 row affected)

Completion time: 2020-04-24T17:29:37.2942924+03:00
```

Οπότε φαίνεται ότι τα διαβάσματα έχουν πέσει αισθητά.

Ας εξετάσουμε και το αντίστοιχο excecution plan



Οπότε όπως παρατηρούμε και από το execution plan γίνεται ένα index seek στον πίνακα sterms,ένα index seek στον πίνακα bibterms,ένα indexseek στον πίνακα bibrecs,ένα indexseek στον πίνακα bibauthors και ένα indexseek στον πίνακα authors.

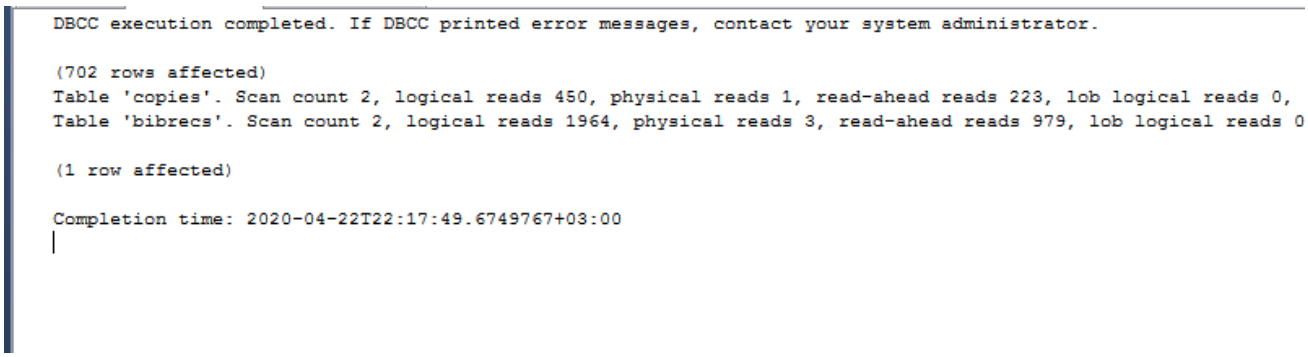
Παρατηρούμε ότι για τον πίνακα authors δεν έχει δημιουργηθεί κάποιο index οπότε πως γίνεται να κάνει indexseek;
Ο λόγος είναι ότι κάθε φορά που ορίζεται primary key σε έναν πίνακα δημιουργείται αυτόματα ενας clustered index.Στην περίπτωσή μας το author έχει primary key το aid.Οπότε ο πίνακας author έχει έναν default clustered index στην στήλη aid χωρίς να τον έχουμε ορίσει εμείς.
Έτσι με την εντολή bibauthors.aid=authors.aid ενεργοποιείται ο default clustered index,οπότε πραγματοποιείται αυτόματα ένα clustered index scan.

Ζήτημα 3:

1ος Κώδικας

```
select distinct bibrecs.bibno,title
from bibrecs,copies
where copyloc='OPA' AND bibrecs.bibno=copies.bibno AND
material='book'INTERSECT(
select bibrecs.bibno,title
from bibrecs,copies
where copyloc='ANA'AND bibrecs.bibno=copies.bibno and
material='book'
)
```

Χωρίς index,ο παραπάνω κώδικας κάνει τα παρακάτω διαβάσματα:



2ος Κώδικας

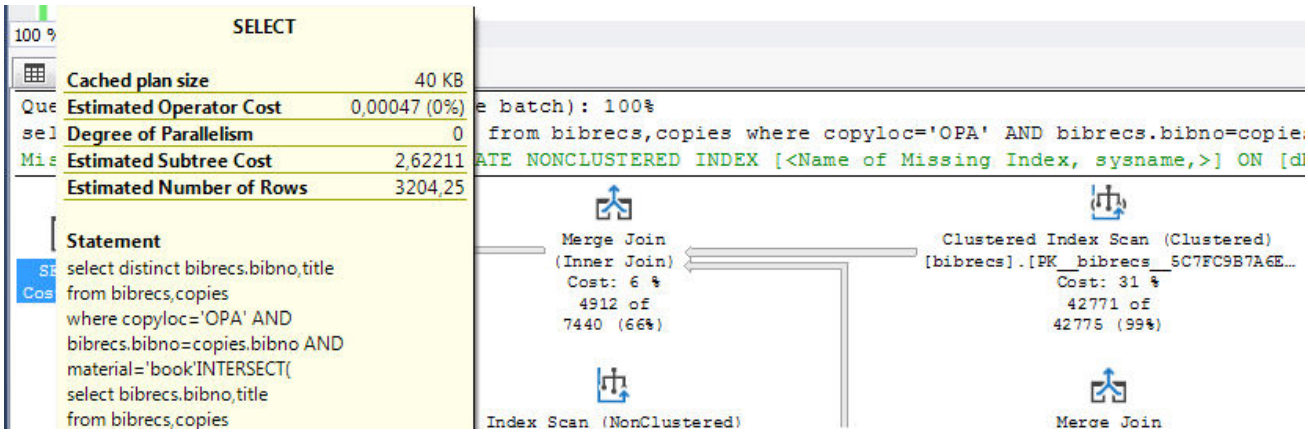
```
Select distinct bibrecs.bibno,title
from bibrecs,copies,copies as c1, copies as c2
where bibrecs.bibno=c1.bibno and bibrecs.bibno=c2.bibno
and c1.copyloc='OPA'and c2.copyloc='ANA'and
material='book'
```

Χωρίς την δημιουργία index ο παραπάνω κώδικας κάνει τα εξής διαβάσματα

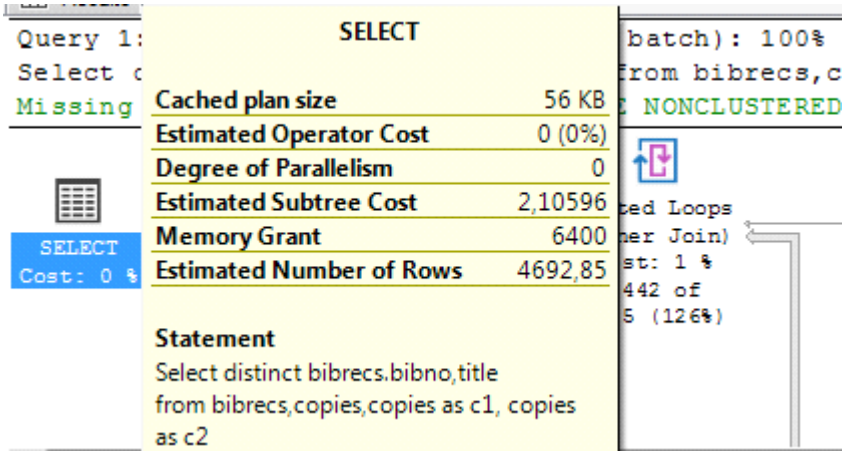
```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(702 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, l
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0,
Table 'copies'. Scan count 3, logical reads 262, physical reads 3, read-ahead reads 255, lob logical reads 0,
Table 'bibrecs'. Scan count 1, logical reads 983, physical reads 3, read-ahead reads 979, lob logical reads 0

(1 row affected)

Completion time: 2020-04-24T17:56:22.1540852+03:00
```

Από αυτούς τους δύο κώδικες επιλέγω τον δεύτερο κώδικα. Όπως φαίνεται και από τις παραπάνω εικόνες ο 2ος κώδικας είναι ταχύτερος από τον πρώτο.Μπορούμε να το καταλάβουμε από τα reads.
Ενναλακτικά μπορούμε να αξιοποιήσουμε και το execution plan ως εξης:
Ελέγχουμε το 'estimated subtree cost' ανάμεσα στους δύο πίνακες και όποιο query έχει το χαμηλότερο subtree cost τότε είναι και το ταχύτερο

EXECUTION PLAN 1ου ΚΩΔΙΚΑ



EXECUTION PLAN 2ου ΚΩΔΙΚΑ



Οπότε παρατηρούμε ότι ο 2ος κώδικας έχει το χαμηλότερο subtree cost(2,10596) έναντι του 1ου κώδικα(2,62211)

2)Τα indexes που θα επιτάχυναν την εκτέλεση του επιλεχθέντος κώδικα(2ου κώδικα στο προηγούμενο ερώτημα)είναι:

1)
```
CREATE INDEX [copy]
ON [dbo].[copies] ([copyloc]) INCLUDE ([bibno])
```

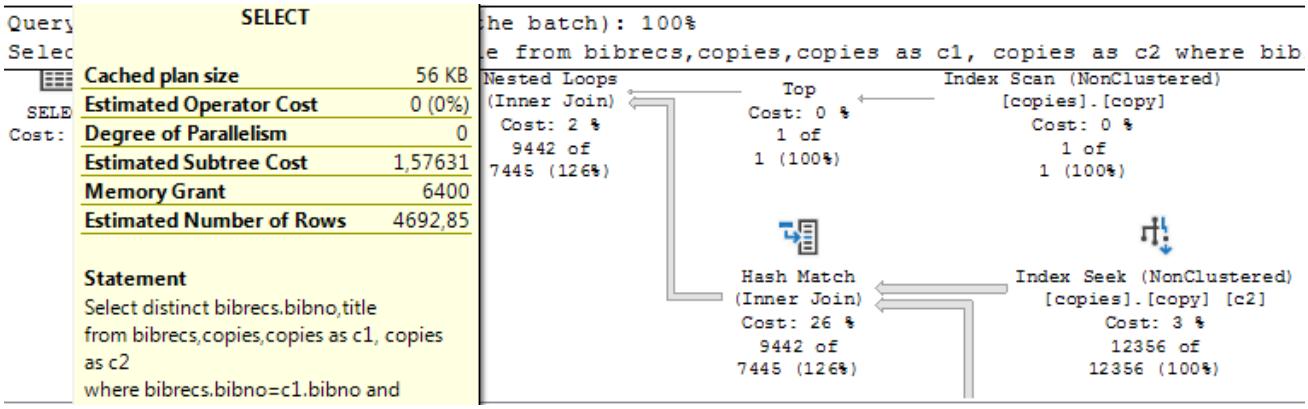
2)
```
CREATE INDEX [indexbib]
ON [bibrecs] ([material])
INCLUDE ([bibno],[title])
```

Πράγματι αν τρέξουμε τώρα τον κώδικα μαζί με τα αντίστοιχα indexes παίρνουμε ότι:

```
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

(702 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0,
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0,
Table 'copies'. Scan count 3, logical reads 262, physical reads 3, read-ahead reads 255, lob logical reads 0
Table 'bibrecs'. Scan count 1, logical reads 300, physical reads 2, read-ahead reads 297, lob logical reads

(1 row affected)

Completion time: 2020-04-22T22:28:48.6176661+03:00
```

Οπότε όπως παρατηρούμε τα reads εχουν πέσει σε σχέση με το αρχικό μου query.

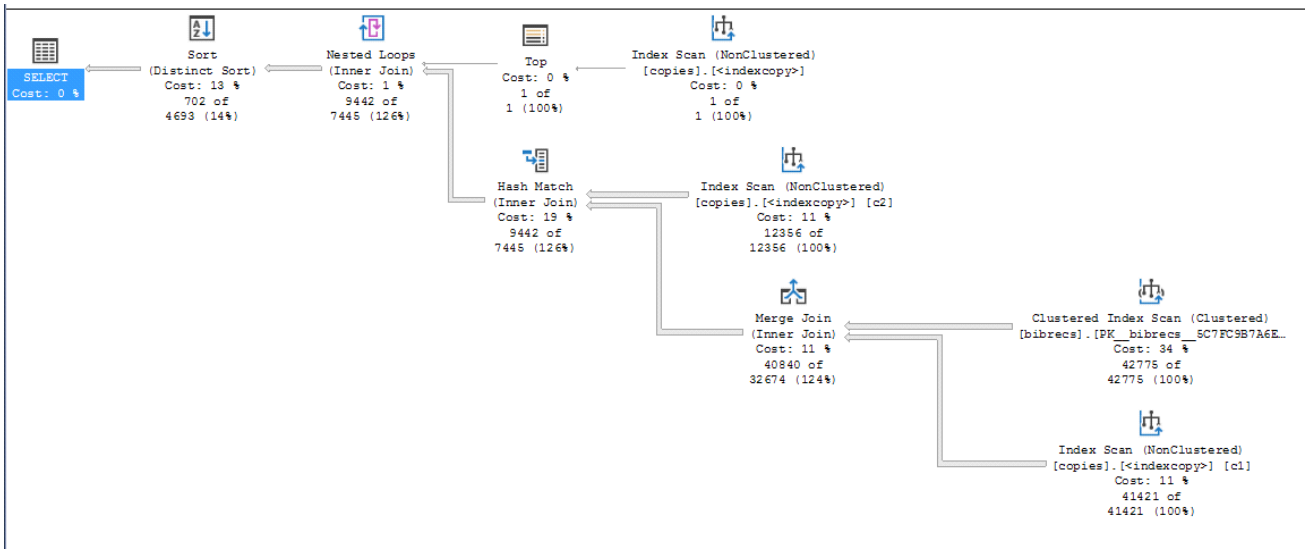
Άλλος τρόπος ελέγχου είναι να ελέξουμε το estimated subtree cost:



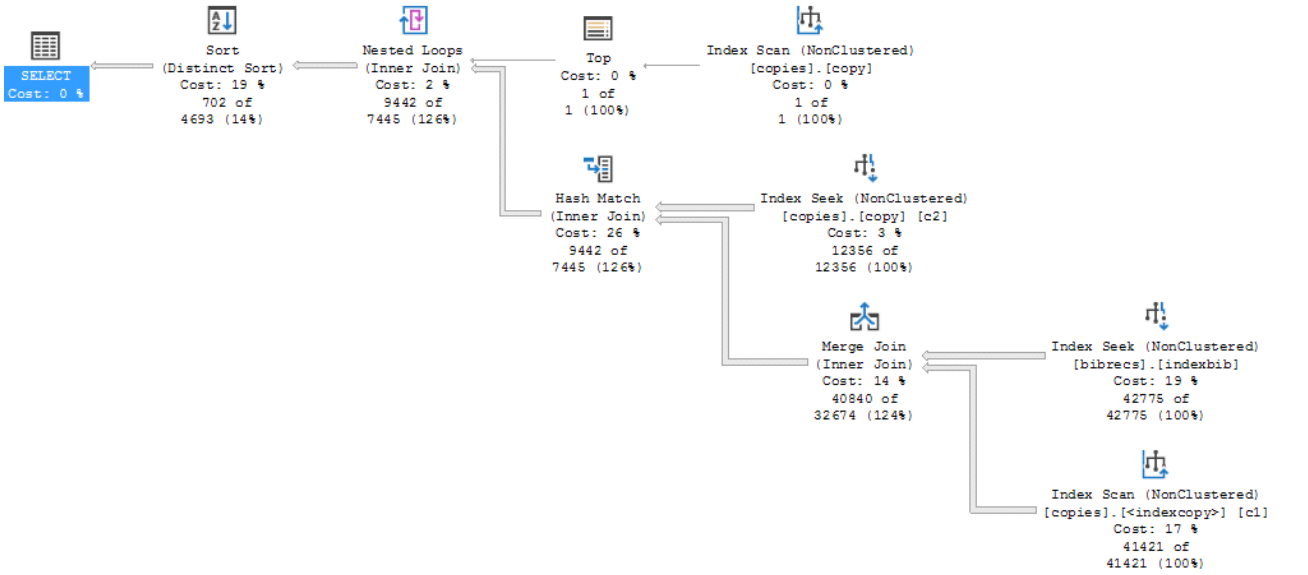
Οπότε παρατηρούμε οτι το Estimated Subtree Cost έχει μειωθεί

Τέλος ας εξετάσουμε τα execution plans προκειμένου να εξηγήσουμε τον λόγο γιατί γίνεται αυτό:

EXECUTION PLAN ΠΡΙΝ ΤΟ INDEX:



EXECUTION PLAN META TO INDEX:



Οπότε από τις παραπάνω εικόνες παρατηρούμε ότι κερδίζουμε χρόνο διότι:
Με τα indexes επιτάχυνα αυτη τη διαδικασία διότι με το Include που συμπεριλήφθηκε πετυχαίνω το εξής:
Επειδή οι στήλες που υπάρχουν στο query περιλαμβάνονται στο ευρετήριο,μπορούν να εντοπιστούν όλες οι τιμές στηλών στο ευρετήριο χωρίς πρόσβαση στον πίνακα με αποτέλεσμα λιγότερες λειτουργίες.
Αντίθετα πριν τα indexes θα έπρεπε να σκανάρω ολόκληρο τον πίνακα προκειμένου να μπορέσω να βρώ που ικανοποιείται η συνθήκη που μου ζητάει.

Ζήτημα 4:
Δημιουργία Πινάκων:

```
create table words(
wid int primary key,
word varchar(200)
```

```
);

create table bibwards(
bibno int foreign key references bibrecs(bibno),
wid int foreign key references words(wid),
primary key(bibno,wid)
);

INDEX:

create index indexonword on words(word)
```