



## ΕΡΓΑΣΤΗΡΙΟ #5

### External Sorting and Evaluation of Relational Operators

1. (**Exercise 13.1**) Suppose you have a file with 10,000 pages and you have three buffer pages. Answer the following questions for each of these scenarios, assuming that our most general external sorting algorithm is used:
- A file with 10,000 pages and three available buffer pages.
  - A file with 20,000 pages and five available buffer pages.
  - A file with 2,000,000 pages and 17 available buffer pages.

#### Questions

- How many runs will you produce in the first pass?
- How many passes will it take to sort the file completely?
- What is the total I/O cost of sorting the file?
- How many buffer pages do you need to sort the file completely in just two passes?

Απάντηση:

- Στο πρώτο πέρασμα (Πέρασμα 0), παράγονται  $\lceil N/B \rceil$  runs  $B$  σελίδων το κάθε ένα, όπου το  $N$  είναι ο αριθμός σελίδων αρχείου και το  $B$  είναι ο αριθμός των διαθέσιμων σελίδων προσωρινής μνήμης:
  - $\lceil 10000/3 \rceil = 3334$  ταξινομημένα runs.
  - $\lceil 20000/5 \rceil = 4000$  ταξινομημένα runs.
  - $\lceil 2000000/17 \rceil = 117648$  ταξινομημένα runs.
- Ο αριθμός των περασμάτων που χρειάζεται για να ταξινομηθεί το αρχείο τελείως, συμπεριλαμβανομένου του αρχικού περάσματος, είναι  $\lceil \log_{B-1} N1 \rceil + 1$ , όπου  $N1 = \lceil N/B \rceil$  είναι ο αριθμός των runs που παράχθηκαν από το Πέρασμα 0:
  - $\lceil \log_2 3334 \rceil + 1 = 13$  περάσματα.
  - $\lceil \log_4 4000 \rceil + 1 = 7$  περάσματα.
  - $\lceil \log_{16} 117648 \rceil + 1 = 6$  περάσματα.
- Δεδομένου ότι κάθε σελίδα διαβάζεται και γράφεται μιά φορά ανά πέρασμα, ο συνολικός αριθμός I/Os σελίδας για την ταξινόμηση του αρχείου είναι  $2 * N * (\#\text{περασμάτων})$ :
  - $2 * 10000 * 13 = 260000$ .
  - $2 * 20000 * 7 = 280000$ .
  - $2 * 2000000 * 6 = 24000000$ .
- Στο πέρασμα 0, παράγονται  $\lceil N/B \rceil$  runs. Στο πέρασμα 1, πρέπει να είμαστε σε θέση να συγχωνεύσουμε αρκετά runs έτσι ώστε  $B - 1 \geq \lceil N/B \rceil$ . Αυτό υπονοεί ότι το  $B$  πρέπει τουλάχιστον να είναι αρκετά μεγάλο έτσι ώστε να ισχύει η ανισότητα  $B * (B - 1) \geq N$ , η οποία μπορεί να χρησιμοποιηθεί για να γίνει μια υπόθεση για το  $B$ . Εν συνεχεία η εικασία πρέπει να επικυρωθεί με τον έλεγχο της πρώτης ανισότητας. Κατά συνέπεια.:

- a. Με 10000 σελίδες στο αρχείο, το  $B = 101$  ικανοποιεί και τις δυο ανισότητες, με  $B = 100$  δεν ισχύει, οπότε χρειαζόμαστε 101 σελίδες προσωρινής μνήμης.
- b. Με 20000 σελίδες στο αρχείο, το  $B = 142$  ικανοποιεί και τις δυο ανισότητες, με  $B = 141$  δεν ισχύει, οπότε χρειαζόμαστε 142 σελίδες προσωρινής μνήμης.
- c. Με 2000000 σελίδες στο αρχείο, το  $B = 1415$  ικανοποιεί και τις δυο ανισότητες, με  $B = 1414$  δεν ισχύει, οπότε χρειαζόμαστε 1415 σελίδες προσωρινής μνήμης.

2. **(Exercise 14.3)** Consider processing the following SQL projection query:

```
SELECT DISTINCT E.title, E.ename FROM Executives E
```

You are given the following information:

Executives has attributes ename, title, dname, and address; all are string fields of the same length.

The ename attribute is a candidate key.

The relation contains 10,000 pages.

There are 10 buffer pages.

Consider the optimized version of the sorting-based projection algorithm: The initial sorting pass reads the input relation and creates sorted runs of tuples containing only attributes ename and title. Subsequent merging passes eliminate duplicates while merging the initial runs to obtain a single sorted result (as opposed to doing a separate pass to eliminate duplicates from a sorted result containing duplicates).

- i. How many sorted runs are produced in the first pass? What is the average length of these runs? (Assume that memory is utilized well and any available optimization to increase run size is used.) What is the I/O cost of this sorting pass?
- ii. How many additional merge passes are required to compute the final result of the projection query? What is the I/O cost of these additional passes?
- iii.
  - a. Suppose that a clustered B+ tree index on *title* is available. Is this index likely to offer a cheaper alternative to sorting? Would your answer change if the index were unclustered? Would your answer change if the index were a hash index?
  - b. Suppose that a clustered B+ tree index on *ename* is available. Is this index likely to offer a cheaper alternative to sorting? Would your answer change if the index were unclustered? Would your answer change if the index were a hash index?
  - c. Suppose that a clustered B+ tree index on *<ename, title>* is available. Is this index likely to offer a cheaper alternative to sorting? Would your answer change if the index were unclustered? Would your answer change if the index were a hash index?
- iv. Suppose that the query is as follows:  
 SELECT E.title, E.ename FROM Executives E  
 That is, you are not required to do duplicate elimination. How would your answers to the previous questions change?

Απάντηση:

- i. Το πρώτο πέρασμα θα παραγάγει 500 ταξινομημένα runs 10 σελίδων κάθε ένα, που κοστίζει 15000 I/Os. Σημ.: οι ταξινομημένες εγγραφές που παράγονται έχουν το μισό μέγεθος σε σχέση με τις αρχικές εγγραφές!
- ii. Για να **συγχωνευτούν** τα runs που παράχθηκαν απαιτούνται  $\lceil \log_{B-1}[N/B] \rceil = \lceil \log_9 500 \rceil = 3$  περεταίρω περάσματα κοστίζοντας  $2 \cdot 3 \cdot 5000 = 30000$  I/Os.
- iii.
  - a. Η χρησιμοποίηση ενός δείκτη συσταδοποιημένου B+ δέντρου στο title θα μείωνε το κόστος σε μια αναζήτηση, ή 12.500 I/Os. Ένας μη συσταδοποιημένος δείκτης θα μπορούσε ενδεχομένως να κοστίζει περισσότερο από 2500+100.000 (2500 από την ανίχνευση του B+ δέντρου, και 10000 \* πλειάδες ανά σελίδα, την οποία υποθέσαμε ότι είναι 10). Κατά συνέπεια, ο δείκτης δεν θα ήταν φτηνότερος. Εάν πρέπει να χρησιμοποιηθεί hash ο δείκτης θα εξαρτώταν από εάν ο δείκτης είναι συσταδοποιημένος. Σε αυτή την περίπτωση, ο hash δείκτης θα ήταν πιθανώς φτηνότερος.
  - b. Η χρησιμοποίηση ενός δείκτη συσταδοποιημένου B+ δέντρου στο ename θα ήταν φτηνότερη από το να ταξινομήσουμε το αρχείο, δεδομένου ότι το κόστος χρήσης του B+ δέντρου θα ήταν 12.500 I/Os. Δεδομένου ότι το ename είναι ένα υποψήφιο κλειδί, κανένας έλεγχος για διπλά <title, ename> ζευγάρια δεν χρειάζεται να γίνει. Ένας μη συσταδοποιημένος δείκτης θα απαιτούσε 2500 (ανίχνευση του δείκτη) + 10000 \* πλειάδες ανά σελίδα I/Os και έτσι πιθανώς θα ήταν ακριβότερος από το να ταξινομήσουμε το αρχείο.
  - c. Η χρησιμοποίηση ενός δείκτη συσταδοποιημένου B+ δέντρου στο <ename, title> θα ήταν επίσης οικονομικώς πιο αποδοτικός από το να ταξινομήσουμε το αρχείο. Ένα μη συσταδοποιημένο B+ δέντρο πάνω στα ίδια χαρακτηριστικά θα επέτρεπε μια αναζήτηση δείκτης-μόνο, και θα ήταν έτσι εξίσου οικονομικό με το συσταδοποιημένο δείκτη. Αυτή η μέθοδος (συσταδοποιημένο και μη) θα κόστιζε περίπου 5000 I/O.
- iv. Ξέροντας ότι η αποβολή διπλών δεν απαιτείται, μπορούμε απλά να ανιχνεύσουμε τη σχέση και να απορρίψουμε τα ανεπιθύμητα πεδία από κάθε πλειάδα. Αυτό είναι η καλύτερη στρατηγική εκτός από στην περίπτωση που ένας δείκτης (συσταδοποιημένος ή μη συσταδοποιημένος) στο <ename, title> είναι διαθέσιμος. Σε αυτήν την περίπτωση, μπορούμε να κάνουμε μια δείκτης-μόνο ανίχνευση. (Σημειώστε ότι ακόμη και με τη χρήση του DISTINCT, καμιά διπλή τιμή δεν υπάρχει στη απάντηση επειδή το ename είναι ένα υποψήφιο κλειδί. Εντούτοις, ένας τυπικός βελτιστοποιητής δεν είναι πιθανό να το αναγνωρίσει αυτό και να παραλείψει το διπλό βήμα αποβολής διπλών τιμών.)