Fall 2010 CS 3200 Class Project: Milestone 7

The goal for this milestone is to understand better the tradeoffs of using different index structures.

This milestone is to be completed individually (i.e., no teams). You can discuss problems with other students, but you have to create all deliverables yourself from scratch. In particular, it is not allowed to copy somebody else's code or text and modify it.

The report for this milestone is due on Monday, **November 22 at 5pm**. For late submissions you will lose one percentage point per hour after the deadline. This milestone is worth 15% of your overall homework score. Please email the deliverables to both me and Yue. You should receive a confirmation email from either of us. If you have not received a confirmation email within 12 hours after submitting your solution or by the time of the deadline, whichever comes first, you need to email us immediately to make sure we actually received your submission. (Of course, if you submit too close to the deadline, you might receive a confirmation sometime within the next 30-60 minutes after you submitted.) If you need to send multiple files, please create a single zip file. Many other attachments types, in particular rar files, are rejected by the CCIS mail server.

I/O Cost Computations

Consider a relation Employee with attributes Age, Salary, Name, Address, Lot, and so on. We have the following information about this relation:

1. The relation contains exactly 48,000 employees. Each employee is between 18 and 65 (inclusive) years old and can earn between $4000 and $4999 (inclusive) per month. The data is perfectly evenly distributed. More precisely, there is **exactly one** employee with each of the allowed age-salary combinations: (18,4000), (18,4001), (18,4002),…, (18,4999), (19,4000), (19,4001),…, (19,4999),…,…, (65,4000),…, (65,4999).
2. Each disk page can store up to 10 of the employee records.
3. When building an index using Alternative 2 (i.e., each data entry is a <key, rid> pair) on a single search key <Age> or <Salary> only, we can fit up to 100 of the corresponding data entries into a single disk page. Similarly, we can fit up to 100 <key, page-pointer> entries into each non-leaf page of a tree index.
4. When building an index using Alternative 2 on a composite search key like <Age,Salary> or <Salary,Age>, we can fit up to 50 of the corresponding data entries into a single disk page. Similarly, we can fit up to 50 <key, page-pointer> entries into each non-leaf page of a tree index.
5. For a tree index, no matter if it is clustered or not, assume every leaf-level page is exactly 60% occupied. E.g., a leaf page that can hold n entries, actually only contains 0.6n entries.

Compute the *number* of page-I/Os for each of the following queries using each of the following file organizations:

Queries:

Q1: SELECT E.age FROM Employees E WHERE E.age >= 20 AND E.age <= 40

Q2: SELECT E.age, E.salary FROM Employees E WHERE E.age >= 20 and E.age <= 40 AND E.salary = 4600

File organizations:

F1: All data records are stored in a heap file (unsorted!), no index is available.

F2: All data records are stored in a clustered tree index (Alternative 1) on <Age>.

F3: All data records are stored in a heap file. There is also an unclustered tree index (Alt. 2) on <Age>.

F4: All data records are stored in a heap file. There is also an unclustered tree index (Alt. 2) on <Age,Salary>.

F5: All data records are stored in a heap file. There is also an unclustered tree index (Alt. 2) on <Salary,Age>.

Make sure you briefly explain how you derived the I/O cost estimates.

Creating an Index

Consider this query from HW 5:

```
SELECT buildingNum, COUNT(studentID)
FROM Student
GROUP BY buildingNum
```

Enter the query in the query composition window and then select Query -> Display Estimated Execution Plan from the main window. Include a sceenshot of the displayed plan in your report.

Then create a *clustered tree index* on table Student on composite search key <buildingNum,studentID>. Include the corresponding index creation code in your report.

Now ask the DBMS again to show you the estimated execution plan. Include a screenshot of this new plan.

Discuss how the two plans are different (or not), and explain briefly what you think the reason is for the DBMS to use or not use the newly created index for the query. Try to interpret the meaning of the displayed plans based on what we discussed in class and maybe search the help documentation or the Web.