University of Edinburgh
School of Informatics
INFR11199 - Advanced Database Systems (Spring 2024)

Tutorial Sheet 3

1. (Sorting and Hashing) Suppose the size of a page is 4 KB, and the size of the memory buffer is 1 MB (1024 KB).

   (a) We have a relation of size 800 KB. How many page I/Os are required to sort this relation and write the sorted relation back to disk?

   (b) We have a relation of size 5000 KB. How many page I/Os are required to sort this relation and write the sorted relation back to disk?

   (c) What is the size of the largest relation that would need two passes to sort?

   (d) What is the size of the largest relation we can possibly hash in two passes (i.e., with just one partitioning phase)?

   (e) Suppose we have a relation of size 3000 KB. We are executing a `DISTINCT` query on a column `age`, which has only two distinct values, evenly distributed. Would sorting or hashing be better here, and why?

   (f) Now suppose we were executing a `GROUP BY` on `age` instead. Would sorting or hashing be better here, and why?

2. (Joins) Consider the following database of students and assignment submissions and the SQL query:

```
CREATE TABLE Students (
  student_id INTEGER PRIMARY KEY,
  ...
);
CREATE TABLE Assignments(
  assignment_number INTEGER,
  student_id INTEGER REFERENCES Students(student_id),
  ...
);
SELECT *
  FROM Students, Assignments
 WHERE Students.student_id = Assignments.student_id;
```

Assume the following:

- `Students` has 20 pages, with 200 records per page
- `Assignments` has 40 pages, with 250 records per page.

(a) What is the I/O cost of a simple nested loop join for `Students` $\bowtie$ `Assignments`?

(b) What is the I/O cost of a simple nested loop join for `Assignments` $\bowtie$ `Students`?

(c) What is the I/O cost of a block nested loop join for `Students` $\bowtie$ `Assignments`? Assume our buffer size is $B = 12$ pages.

(d) What is the I/O cost of a block nested loop join for `Assignments` $\bowtie$ `Students`? Assume our buffer size is $B = 12$ pages.

(e) What is the I/O cost of an Index-Nested Loop Join for `Students` on $\bowtie$ `Assignments`?

Assume we have a *clustered* variant B index on `Assignments.student_id`, in the form of a height 2 B+ tree. Assume that: index (non-leaf) nodes and leaf pages are not cached; all hits are on the same leaf page; and all hits are also on the same data page. only one matching page

(f) Now assume we have an *unclustered* variant B index on `Assignments.student_id`, in the form of a height 2 B+ tree. Assume that index node pages and leaf pages are never cached, and we only need to read the relevant leaf page once for each record of `Students`, and all hits are on the same leaf page.

What is the I/O cost of an Index-Nested Loop Join for `Students` $\bowtie$ `Assignments`?

Hint: The foreign key in `Assignments` may play a role in how many accesses we do per record. no info on total number of matching tuples

(g) What is the cost of an *unoptimized* sort-merge join for `Students` $\bowtie$ `Assignments`? Assume we have $B = 12$ buffer pages.

(h) What is the cost of an *optimized* sort-merge join for `Students` $\bowtie$ `Assignments`? Assume we have $B = 12$ buffer pages.

(i) In the previous question, we had a buffer of $B = 12$ pages. If we shrank B enough, the answer we got might change.
How small can the buffer B be without changing the I/O cost answer we got?

(j) What is the I/O cost of Grace Hash Join on these tables?
Assume uniform hash partitioning and a buffer pool consisting of $B = 6$ pages.