Ace Cassidy CPTS451 HW5 Spring'20

Question 1: Storage and Indexing

Schema Sailors (sid: integer, sname: string, rating: integer, age: integer, address: string)

- The sid is a key (i.e., sid values are unique).
- Assume sid values are uniformly distributed between 1 and 100,000
- Each tuple of Sailors is 50 bytes long.
- The relation contains 80,000 tuples.
- Each page can hold 50 Sailors tuples.
- Assume the time to read/write to/from a page is D; assume the records are compacted and there is no gap between records.
- Assume 1KB= 1000 bytes

A. Assume relation Sailors is stored in a heap file. What is the estimated cost for each

1. file scan

   BD = (80,000 / 50)D = 1600D This One read per page

2. equality search

   0.5 x BD = 0.5 x 1600 x D = 800D This Same as file scan but on average only have to look at half of tile

3. range search

   BD = 1600D This Have to scan all pages since there is no ordering

B. Assume there is a clustered B+ tree index on sid using alternative-1 for relation Sailors. Assume the B+tree has 67% occupancy, i.e., the physical data pages are 1.5 times more than original data file. Assume the height of the B+tree is 4

1. file scan

   1.5 x BD = 2400D

2. equality search

   height x D = 4D

3. range search (20,000 <= sid<40,000)

   file scan x percentage of pages + height x D This 2400 x (20000 / 100000) x D + 4D This 484D

C. Assume there is an unclustered B+ tree index on sid using alternative-3 for relation Sailors. Assume that data entry size for the unclustered B+tree index is 1/10th (i.e., 10%) of the actual tuplesize. Assume the B+tree has 67% occupancy, i.e., the index pages are 1.5 times more than sequential index. Assume the height of the B+tree is 4

1. file scan

   1600D

2. equality search(sid=20,000)

   tree search + index page This 4D + 1

3. range search (20,000 <= sid<40,000)

   Cost to search tree + Cost of reading index pages + Cost of reading entriesThis 4D + (80,000 / (50 x 10)) x 1.5 x .2 + 20,000 x .2 This 4D + 48D + 16,000D

Question 2: Indexing

Schema Prof(ssno, pname, office, age, rank, specialty, dept_did)

Dept(did, dname, budget, num_majors, chair_ssno)

- ssno is the primary key for Prof and did is the primary key for Dept.
- Prof.dept_did is a foreign key referencing Dept.did
- Each professor is involved with some department.

Suppose you know that the following queries are the seven most common queries in the workload for this university and all seven are roughly equivalent in frequency and importance

Query 1: List the names, ages, and offices of professors of a user-specified rank (e.g. Associate Professor) who have a user-specified research specialty (e.g., artificial intelligence). Assume that the university has a diverse set of faculty members, making it very uncommon for more than a few professors to have the same research specialty an unclustered hash index on rank and specialty for the Prof relation. This allows for fast file scan with rank and specialty grouped together

Query 2: List all the information for professors in a user specified age range This a clustered B+ tree on age for the Prof relation. This allows for fast range search for range queries due to ordering by age

Query 3: List the department id, department name, and chairperson name for departments with a user-specified number of majors an unclustered hash index on num_majors for the Dept relation This allows for fast file scan with num_majors grouped together

Query 4: List all the information about professors who are department chairpersons an unclustered hash index on chair_ssno for the Dept relation and an unclustered B+ tree index on ssno for the Prof relation Thisfast file scan on Dept grouped by chair_ssno and fast equality search of Prof based on ssno

Query 5: List the "did" of each department and the number of professors with "Associate Professor" rank in that department an unclustered hash index on did for the Dept relation and an unclustered hash on rank and dept_did for the Prof relation Thisfast file scan on Dept and fast equality search on dept_did

Query 6: Find the department(s) with the fewest number of majors a clustered B+ tree on num_majors for the Dept relation This ordering by num_majors allows for finding minimum quickly

Query 7: Find the youngest professor who is a department chairperson an unclustered hash index on chair_ssno for the Dept relation and a clustered B+ tree index on ssno and age for the Prof relation. A hash on chair_ssno allows for fast file scan, and ordering by ssno and age allows for finding minimum quickly