Washington State University

School of Electrical Engineering and Computer Science

CptS 451 – Introduction to Database Systems

Dr. Sakire Arslan Ay

Homework-6

Due Date: Wednesday, April 29, 11:59pm

Name: _____

Student Number:_____

Question:	Max points:	Score:
1	30	
2	70	
Total	100	

(30pts) Question 1: Storage and Indexing

Consider the relation below:

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

- The s*id* 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) **(10pts)** Assume relation Sailors is stored in a heap file. What is the estimated cost of (i) file scan, (ii) equality search (sid=20,000), (iii) range search (20,000 <= sid < 40,000) on Sailors?

(b) **(10pts)** Assume there is a clustered B+ tree index on *sid* using alternative-1 for relation Sailors. What is the cost of (i) file scan, (ii) equality search (sid=20,000), (iii) range search (20,000 <= sid < 40,000) on 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.

(c) **(10pts)** Assume there is an unclustered B+ tree index on *sid* using alternative-3 for relation Sailors. What is the cost of (i) file scan , (ii) equality search (sid=20,000), (iii) range search (20,000 <= sid < 40,000) on Sailors?

- Assume that data entry size for the unclustered B+tree index is 1/10 th (i.e., 10%) of the actual tuple size.
- 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.

For each of the questions above, show how you obtained the answer.

(70pts) Question2: Indexing

Consider the following relational schema for a portion of a university database:
Prof(ssno, pname, office, age, rank, specialty, dept_did)
Dept(did, dname, budget, num_majors, chair_ssno)

Note that:
- 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:

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.
2. List all the information for professors in a user specified age range.
3. List the department id, department name, and chairperson name for departments with a user-specified number of majors.
4. List all the information about professors who are department chairpersons.
5. List the "did" of each department and the number of professors with "Associate Professor" rank in that department.
6. Find the department(s) with the fewest number of majors.
7. Find the youngest professor who is a department chairperson.

These queries occur much more frequently than updates, so you should build whatever indexes you need to speed up these queries. However, you should not build any unnecessary indexes (or include any unnecessary attributes in an index), as updates will occur (and would be slowed down by unnecessary indexes). Given this information, decide which attributes should be indexed and whether each index should be a clustered index or an unclustered index. Assume that both B+ trees and hashed indexes are supported by the DBMS, and that both single- and multiple-attribute index keys are permitted.

For each index:
- identify the attributes you recommend indexing on (you can propose to create one or more new indexes for each query or you may suggest to re-use the indexes that are already created (proposed) for other queries.)
- indicate whether each index should be clustered or unclustered, and
- whether it should be a B+ tree or a hashed index.
- briefly describe how that index will be used to answer the query.

Submission Instructions:

HW6 will be submitted on Blackboard. You may either type it or handwrite and scan it as pdf (only neat and readable handwriting please).