Exercise 4 – INLS 623

Ramakrishnan Chapter 20, Exercise 20.1.1 and 20.1.2 except the last bullet in 20.1.2 (Page 685)
Consider the following BCNF schema and a portion of a simple corporate database (type information is not relevant to this question and is omitted):

Emp(*eid, ename, addr, sal, age, yrs, deptid*)
Dept(*did, dname,floor, budget*)

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

- List the id, name and address of employees in a user-specified age range
- List the id, name and address of employees who work in the department with a user-specified department name
- List the id and address of employees with a user- specified employee name
- List the overall average salary for employees
- List the average salary for employees of each age; that is, for each age in the database, list the age and the corresponding average salary.
- List all the department information, ordered by department floor numbers.

1. Given the information and assuming that these queries are more important than any updates, design a physical schema for the corporate database that will give good performance for the expected workload. Specify first the physical design you would choose for each individual query above. Then determine an overall choice of indexes for your physical design taking all the queries into account. In all cases, decide which attributes on each table should be indexed, whether the index is dense or non-dense, note if you are setting it up for an index only solution, and be sure to identify what attribute you are ordering the file by (your "*one free index*", if you choose to do so). Assume that both B+ trees and hashed indexes are supported by the DBMS and that both single- and multiple-attribute index search keys are permitted. When you choose your overall choice, design a physical schema for the university database that will give good performance for the expected workload, but you should not build any unnecessary indexes, as updates will occur (and would be slowed down by unnecessary indexes). In all cases explain you rationale.

2. Redesign the physical schema assuming that the set of important queries is changed to be the following:
   - List the id and address of employees with user-specified employee name
   - List the overall maximum salary for employees

- List the average salary for employees by department, ie., for each *deptid* value, list the *deptid* value and the average salary of employees in that department.
- List the sum of the budgets of all departments by floor; that is, for each floor, list the floor and the sum.

Prepare your answer for each query (what's the best index solution for that query) and prepare an overall answer that gives the best overall plan for index choice(s) for the tables. You can also use either a Hash or B+Tree file ordering for each table (but if you choose a Hash ordering you get only that single Hash ordering, i.e. no other Hash or B+Tree indexes on that table). For your output you should list the

- SQL equivalent to the query
- Description of your how you think the query will be processed and how you choose your index(es).
- Ordering (or not) of each table file.
- Specify each index for each table, include (B+Tree or Hash, non-dense vs dense, whether it's index only)

Example listing, showing what information is expected when spelling out indexes:

TEMPLATE for Unordered:
Table XXX is not ordered because….

TEMPLATE for Hash:
Order table XXX by HASH on YYY
Secondary Dense Index on XXX(AAA), can do, but can be complicated and depends on DBMS
Because: ….

TEMPLATE for B+Tree:
Order table XXX by B+Tree on YYY
Non-dense B+ Tree primary index on YYY
Secondary Dense Index on XXX(AAA)
Secondary Dense Index on XXX(BBB)
Secondary Dense Index on XXX(CCC), as INDEX ONLY solution for ZZZ
Because: ….

EXAMPLE:
Dept table:
order table file by dept(dname)
Non-dense B+ tree on dept(dname)
Dense B+ tree index on dept(salary)
Dense B+ tree index on dept(ename, deptid)

Emp table: order table file by emp(eid)
Non-dense B+ tree on emp (eid)

Dense Secondary B+ tree index on emp (salary)
Dense Secondary B+ tree index on emp (ename, deptid)

WorksOn table: order file by hash WorksOn(eid)

Projects table: Not ordered

Explanation: XXXX