# 1 Query Optimization

Consider a relation with this schema:

Employees(*eid* : **integer**, *ename* : **string**, *sal* : **integer**, *title* : **string**, *age* : **integer**)

Suppose that the following indexes, all using indirect storage (i.e., an index entry is a pointer to the record where the data is) for data entries, exist: a hash index on *eid*, a B+ tree index on *sal*, a hash index on *age*, and a clustered B+-tree index on $<age, sal>$. Each Employees record is 100 bytes long, and you can assume that each index data entry is 20 bytes long. The Employees relation contains 10,000 pages.

1. Consider each of the following selection conditions and, assuming that the reduction factor (RF) for each term that matches an index is 0.1, compute the cost of the most selective access path for retrieving all Employees tuples that satisfy the condition:

   **(a)** $sal > 100$
   **(b)** $age = 25$
   **(c)** $age > 20$
   **(d)** $eid = 1,000$
   **(e)** $sal > 200 \wedge age > 30$
   **(f)** $sal > 200 \wedge age = 20$
   **(g)** $sal > 200 \wedge title =' CFO'$
   **(h)** $sal > 200 \wedge age > 30 \wedge title =' CFO'$

2. Suppose that, for each of the preceding selection conditions, you want to retrieve the average salary of qualifying tuples. For each selection condition, describe the least expensive evaluation method and state its cost.

3. Suppose that, for each of the preceding selection conditions, you want to compute the average salary for each age group. For each selection condition, describe the least expensive evaluation method and state its cost.

4. Suppose that, for each of the preceding selection conditions, you want to compute the average age for each sal level (i.e., group by sal). For each selection condition, describe the least expensive evaluation method and state its cost.

5. For each of the following selection conditions, describe the best evaluation method:

   **(a)** $sal > 200 \vee age = 20$
   **(b)** $sal > 200 \vee title =' CFO'$
   **(c)** $title =' CFO' \vee ename =' Joe'$

# 2 Query Optimization

Consider the following relational schema and SQL query. The schema captures information about employees, departments, and company finances (organized on a per department basis).

Emp(*eid* : **integer**, *did* : **integer**, *sal* : **integer**, *hobby* : **char(20)**)

Dept(*did* : **integer**, *dname* : **char(20)**, *floor* : **integer**, *phone* : **char(10)**)

Finance(*did* : **integer**, *budget* : **real**, *sales* : **real**, *expenses* : **real**)

Consider the following query:

SELECT D.dname, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.did=D.did AND D.did=F.did AND D.floor=1
AND E.sal $\geq$ 59000 AND E.hobby = 'yodeling'

1. Identify a relational algebra tree (or a relational algebra expression if you prefer) that reflects the order of operations a decent query optimizer would choose.
2. List the join orders (i.e., orders in which pairs of relations can be joined to compute the query result) that a relational query optimizer will consider. (Assume that the optimizer follows the heuristic of never considering plans that require the computation of cross-products.) Briefly explain how you arrived at your list.
3. Suppose that the following additional information is available: Unclustered B+ tree indexes exist on *Emp.did*, *Emp.sal*, *Dept.floor*, *Dept.did*, and *Finance.did*. The system's statistics indicate that employee salaries range from 10,000 to 60,000, employees enjoy 200 different hobbies, and the company owns two floors in the building. There are a total of 50,000 employees and 5,000 departments (each with corresponding financial information) in the database. The DBMS used by the company has just one join method available, index nested loops.
   **(a)** For each of the query's base relations (Emp, Dept, and Finance) estimate the number of tuples that would be initially selected from that relation if all of the non-join predicates on that relation were applied to it before any join processing begins.
   **(b)** Given your answer to the preceding question, which of the join orders considered by the optimizer has the lowest estimated cost?

# 3 Concurrency Control

Read the article "Concurrency Control and Recovery" by Mike Franklin (you can find it at http://infosys.cs.uni-saarland.de/teaching/dbs0809/exercises/franklin.pdf).

1. Give examples of one serializable and one non-serializable schedule and create precedence graphs for them. Also point out the conflicting pairs.
2. Explain the differences between STEAL and NO-STEAL and between FORCE and NO-FORCE policies.
3. Explain the differences between the "basic" 2PL and strict 2PL.
4. What is the phantom problem? Given an example of how it could occur and how could it be avoided.

# 4 Strict 2PL

Show how the following transaction histories would be handled by a scheduler based on strict 2PL. Add information to the transaction histories regarding:

- which locks are acquired (and when);
- whether any deadlock is incurred at any point;
- what is the waits-for graph for each deadlock situation;
- which transactions are aborted in the event of a deadlock;
- which transactions succeed.

**(a)** R1(A) R2(A) W1(A) W2(A) C1 C2

**(b)** W3(A) R1(A) W1(Z) R2(B) W2(Y) W3(B) C1 C2 C3

**(c)** R1(A) R2(B) R3(A) W1(A) R3(B) W3(B) A2 R3(C) W3(C) C3 C1