Chapter 3: Algorithms for Query Processing and Optimization

Question 3.1. List and describe typical steps when a query is processed.

Question 3.2. Differentiate a query tree from a query graph.

Question 3.3. Why does a SQL query need to be translated into relational algebra expressions?

Question 3.4. Describe external sorting and calculate its cost. List some applications of sorting in query processing.

Question 3.5. A file of 4096 blocks is to be sorted with an available buffer space of 64 blocks. How many passes will be needed in the merge phase of the external sort-merge algorithm?

Question 3.6. How are SELECT operations implemented? Give an example.

Question 3.7. How are JOIN operations implemented? Give an example.

Question 3.8. How are PROJECT operations implemented? Give an example.

Question 3.9. How are aggregate operations implemented? Give an example.

Question 3.10. How are SET operations implemented?

Question 3.11. Given queries as follows, for each query, write its corresponding SQL statement, draw its query tree, and then explain its processing to obtain the result.

3.11.1. Retrieve the last name and salary of each employee who works in department 10 and has a salary higher than 30,000.

3.11.2. Retrieve the last name and department number of each employee who works in the department where the minimum salary of the employees is higher than 30,000.

3.11.3. Retrieve the department name and department number of each department where more than 10 employees work with a salary higher than 30,000.

3.11.4. For each department, retrieve its name and the number of employees who work for the department with a salary higher than 30,000.

3.11.5. Retrieve the name and address of employees who work for the 'Research' department.

3.11.6. Retrieve the name and the department name of each employee who works for department 5 with a salary from 20,000 to 50,000.

3.11.7. Retrieve the department name and department number of each department where there exists an employee with a salary higher than 30,000.

3.11.8. Retrieve the names of employees whose salary is greater than the salary of all the employees in department 5.

3.11.9. For each department that has more than 5 employees, retrieve the department number and the number of its employees who are making more than 40,000.

3.11.10. For each employee who works for the department that has more than 10 employees, retrieve the employee name and department name if he/she has a salary higher than 30,000.

Question 3.12. What is an execution plan? Give an example of a query and its execution plan.

Question 3.13. What is a heuristic optimizer? What are its heuristic rules?

Question 3.14. What is a cost-based optimizer? How is it different from a heuristic optimizer?

Question 3.15. Describe cost components for a cost function to estimate a query execution cost. What kind of databases uses each cost component?

Question 3.16. Differentiate pipelining from materialization. Demonstrate their differences.

Question 3.17. Given the three following relations:

Supplier(Supp#, Name, City, Specialty)

Project(Proj#, Name, City, Budget)

Order(Supp#, Proj#, Part-name, Quantity, Cost)

and a SQL query:

```
SELECT Supplier.Name, Project.Name
FROM Supplier, Order, Project
WHERE Supplier.City = 'New York City' AND Project.Budget > 10000000 AND
      Supplier.Supp# = Order.Supp# AND Order.Proj# = Project.Proj#;
```

3.17.1. Write the relational algebraic expression that is equivalent to the above query and draw a query tree for the expression.

3.17.2. Apply the heuristic optimization transformation rules to find an efficient query execution plan for the above query. Assume that the number of the supliers in New York is larger that the number of the projects with the budgets more than 10000000$.

Question 3.18. Draw query trees step by step to obtain a final optimized query tree using heuristic optimization for each query in 3.11.

Question 3.19. Using the characteristics of the EMPLOYEE and DEPARTMENT data files as described below, describe an optimized execution plan based on a decision of the cost-based optimizer for each query in 3.11.

```
CREATE TABLE EMPLOYEE (
        Fname VARCHAR(15) NOT NULL,
        Minit CHAR,
        Lname VARCHAR(15) NOT NULL,
        Ssn CHAR(9) NOT NULL,
        Bdate DATE,
        Address VARCHAR(30),
        Sex CHAR,
        Salary DECIMAL(10,2),
        Super_ssn CHAR(9),
        Dno INT NOT NULL DEFAULT 1,
        PRIMARY KEY (Ssn),
        CONSTRAINT EMPSUPERFK FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn) ON DELETE
        SET NULL ON UPDATE CASCADE,
        CONSTRAINT EMPDEPTFK FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber) ON DELETE
        SET DEFAULT ON UPDATE CASCADE
);
CREATE TABLE DEPARTMENT (
        Dname VARCHAR(15) NOT NULL,
        Dnumber INT NOT NULL,
        Mgr_ssn CHAR(9) NOT NULL,
        Mgr_start_date DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
);
```

The EMPLOYEE file has: $r_E = 10{,}000$, $b_E = 2000$, $bfr_E = 5$, and the following access paths:

- A clustering index on SALARY, with levels $x_{SALARY} = 3$ and average selection cardinality $S_{SALARY} = 20$.
- A secondary index on the key attribute SSN, with $x_{SSN} = 4$ ($S_{SSN} = 1$).
- A secondary index on the nonkey attribute DNO, with $x_{DNO}= 2$ and first-level index blocks $b_{I1DNO}= 4$. There are $d_{DNO} = 125$ distinct values for DNO, so the selection cardinality of DNO is $S_{DNO} = \lceil r_E/d_{DNO} \rceil = 80$.
- A secondary index on SEX, with $x_{SEX} = 1$. There are $d_{SEX} = 2$ values for the sex attribute, so the average selection cardinality is $S_{SEX} = \lceil r_E/d_{SEX} \rceil = 5000$.

The DEPARTMENT file has: $r_D = 125$ and $b_D = 13$, and the following access paths:

- A primary index on DNUMBER, with levels $x_{DNUMBER} = 1$.
- A secondary index on MGRSSN of DEPARTMENT, with $s_{MGRSSN} = 1$, $x_{MGRSSN} = 2$.

$js_{EMPLOYEE\text{-}DEPARTMENT\ on\ DNO=DNUMBER} = (1/|DEPARTMENT|) = 1/r_D = 1/125$, $bfr_{ED} = 4$