ADBMS Assignment-I Answers

1. Disk File Problem:

Given:

Number of records = 28,000

Block size = 256 bytes

Record size = 111 bytes

(a) No. of blocks needed to store the file:

Blocking Factor (bfr) = Block size   Record size = 256   111 ≈ 2 records per block

Total Blocks needed = Number of records   bfr = 28,000   2 = 14,000 blocks

(b) No. of Sorted Runs and Passes (Epternal Sorting):

Memory Buffers Available (M) = 11 blocks

Initial runs = 14,000   11 ≈ 1273 runs

Number of passes = 1 + log(M-1)(Number of runs) ≈ 4 passes

(c) No. of Disk Block Accesses:

Each pass = 2 p 14,000 = 28,000 accesses

For 4 passes: Total Accesses = 4 p 28,000 = 112,000 block accesses

2. Heuristic Query Optimization:

Query 1 Optimization:

Apply selections early: σ(D.department_name = "Computer Science") (D) and σ(S.semester = "Fall99") (S)

Join S and D

Project section_id

Query 2 Optimization:

Apply selection: σ(PLOCATION = 'Stafford') (PROJECT)

Join PROJECT and DEPARTMENT on DNUM = DNUMBER

Join with EMPLOYEE on MGRSSN = SSN

Project required fields

3. Fine-grain vs Coarse-grain Parallel Machines:

Fine-grain Parallelism Coarse-grain Parallelism

Very small tasks Larger tasks

High communication overhead Less communication overhead

Suitable for SIMD systems Suitable for MIMD systems

4. Performance of a Parallel Machine:

Measured using Speedup, Efficiency, Execution time, Throughput, Scaleup.

5. Speedup and Scaleup:

Speedup (S) = Time (one processor) Time (multiple processors)

Scaleup = Handling bigger problems with more processors at the same time.

6. Blocking Factor (New Problem):

Given:

Records = 15,000

Block size = 512 bytes

Record size = 110 bytes

Calculations:

Blocking Factor = 512   110 ≈ 4

Blocks needed = 15,000   4 = 3750 blocks

Index File calculation:

Attribute size = 7 bytes

Pointer size = 9 bytes

Total per index entry = 16 bytes

Blocking Factor of index = 512   16 = 32 entries per block

Number of index blocks = 3750   32 ≈ 118 blocks

7. Relational Algebra Expression:

SQL Query:

SELECT item_name

FROM ITEM A, SALES B, LOCATION C

WHERE A.Itemno=B.Itemno AND B.loc_id=C.loc_id AND

LocationName="Delhi" AND A.Itemprice>10000;

Relational Algebra:

σ(LocationName="Delhi" AND Itemprice>10000) ((ITEM ⋈ A.Itemno=B.Itemno SALES) ⋈ B.loc_id=C.loc_id LOCATION)

Then project π(item_name).

8. SQL Statements:

(a) Booknames never borrowed:

```
SELECT BName FROM Book WHERE Accno NOT IN (SELECT Accno
FROM Transaction);
```

(b) Accno, Bookname, and number of copies:

```
SELECT Accno, BName, COUNT(*) AS NoOfcopies FROM Book GROUP By
Accno, BName;
```

(c) Borrower with Booktype 'Journal':

```
SELECT DISTINCT Borrower.BorrName FROM Borrower, Transaction,
Book WHERE Borrower.Borrowerno = Transaction.Borrowrno AND
Transaction.Accno = Book.Accno AND Book.Type = 'Journal';
```

(d) Borrowers keeping book more than 30 days:

```
SELECT DISTINCT BorrName FROM Borrower, Transaction WHERE
Borrower.Borrowerno = Transaction.Borrowrno AND
DATEDIFF(CURDATE(), issuedate) > 30;
```

(e) Borrowers who never borrowed:

```
SELECT BorrName FROM Borrower WHERE Borrowerno NOT IN (SELECT
DISTINCT Borrowrno FROM Transaction);
```

9. Armstrong's Inference Rules:

Reflexivity: If $y \subseteq x$, then $x \to y$

Augmentation: If $x \to y$, then $xz \to yz$

Transitivity: If $x \to y$ and $y \to z$, then $x \to z$

Proof Sketch:

Reflexivity: A set implies its subset.

Augmentation: Adding same attributes preserves dependency.

Transitivity: If x implies y and y implies z, then x implies z.

10. Primary vs Secondary Index:

Primary Index	Secondary Index
Built on primary key	Built on non-primary key
One per table	Multiple possible
Records physically sorted	Records not necessarily sorted