Question 1. Consider a movie database with the following schema:

- ACTOR (<u>AID</u>, AName, Gender, DOB)
- MOVIE (<u>MID</u>, MName, Year, Profit)
- ROLE (<u>AID</u>, <u>MID</u>, RoleName, Pay)

ACTOR and MOVIE record information about actors and movies, respectively. Whenever an actor is cast in a move, the ROLE table records the actor's role and pay in the movie. The relation ACTOR has 5,000 tuples and 100 tuples fit into one block; relation MOVIE has 1,000 tuples and 100 tuples fit into one block; the relation ROLE has 100,000 tuples and 100 tuples fit into one block.

Answer the following questions.

(a) Represent the following queries with relational algebra. You may use the following operators:
σ (selection), Π (projection), ∪ (set union), ∩ (set intersection), − (set difference), ← (assignment), ρ (rename), ⋈ (natural join), G (grouping and aggregation)

(i) Find the names of the female actors who have appeared in at least one movie with a profit over 1,000,000.
Solution:
$R_1 \leftarrow \sigma_{Gender='F' \text{ AND Profit} > 1000000}$ (Actor ⋈ Role ⋈ Movie)
Answer ← Π_{AName} (R_1)

(ii) For every male actor, list his AID, the movies that he appeared in from 2 009 to 2019, as well as the total pay he received during this period.
Solution:
$R_1 \leftarrow \sigma_{Year >= 2009 \text{ AND Year} <= 2019 \text{ AND Gender}='M'}$ (Actor ⋈ Role ⋈ Movie)
Answer $\leftarrow_{AID}$ $G_{COUNT(MID), Sum(Pay)}$ (R_1)

(b) Write the SQL query for a.i and a.ii, respectively.

Solution for Q (a.i):

```
Select AName from
ACTOR natural join ROLE natural join MOVIE
where Gender='F' AND Profit>1000000
```

Solution for Q (a.ii):

```
Select AID, Count(MID), Sum(Pay) from
ACTOR natural join ROLE natural join MOVIE
where Gender='M'  AND Year>=2009 AND Year<=2019
group by AID
```

(c) Assume that we use block nested-loop join to join ACTOR and ROLE. Further assume that the block nested loop join includes the following optimization: If the memory has M blocks, we read in $M-2$ blocks of the outer relation at a time, and when we read each block of the inner relation we join it with all the $M-2$ blocks of the outer relation.

Then, if the memory buffer has 4 blocks, which relation should be used as the outer relation so that the IO cost (in terms of the number of block transfers and disk seeks) of the join operation is smaller? If the memory buffer has 60 blocks, what is the IO cost if ACTOR is used as the outer relation? Show your calculation details.

Solution:

Clearly, it can be calculated that relation ACTOR includes 50 blocks and relation ROLE include 1000 blocks.

First, consider the case when the memory buffer has 4 blocks. If ACTOR is used as the outer relation, the number of block transfer is $\lceil\frac{50}{4-2}\rceil \cdot 1000 + 50 = 25050$ and the number of seeks is $2 \cdot \lceil\frac{50}{4-2}\rceil = 50$. However, if we use ROLE as the outer relation, the number of block transfer is $\lceil\frac{10000}{4-2}\rceil \cdot 50 + 1000 = 25100$ and the number of seeks is $2 \cdot \lceil\frac{1000}{4-2}\rceil = 1000$. Therefore, ACTOR should be used as the outer-relation.

Next, consider the case when the memory buffer has 60 blocks and ACTOR is used as the outer relation. the number of block transfer is $\lceil\frac{50}{60-2}\rceil \cdot 1000 + 50 = 1050$ and the number of seeks is 2.

Question 2. Given the following simplified employee database schema and the SQL query as follows. The attributes can be interpreted in a usual sense. For example, eno is a unique employee number, dno is a unique department number and emp.name is almost distinct.

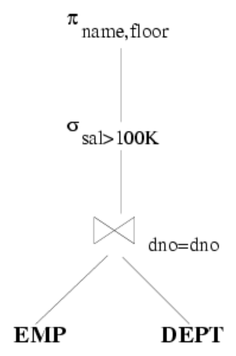
Database schema: EMP(eno,name,dno,sal), DEPT(dno,floor).

SQL: SELECT name, floor

FROM EMP, DEPT

WHERE emp.dno=dept.dno and sal >100K

Consider the following naïve query tree generated for execution of the above SQL query.



We assume there is no index to be used in performing the join operation and use block nested-loop join algorithm to perform the equi-join operation. The data for estimating the access cost of joining the relations EMP and DEPT in the lowest level are as follows:

- Number of tuples of EMP: Ne = 30,000.
- Number of tuples of DEPT: Nd = 1000.
- Number of blocks of EMP: Be = 1000.
- Number of blocks of DEPT: Bd = 200.

(i) Estimate the block access cost of the join operation in the best and worst cases and write down the minimum memory buffer needed in each case.

Solution:

- Worst case = Bd x Be + Bd = 200x 1000 +200=200,200;
  Minimum memory needed = 3 blocks
- Best case = Bd + Be = 1200;
  Minimum memory needed = 200+1+1=202 blocks

(ii) Assuming there is roughly 2% of employees having salary > 100K. Estimate the size of the SQL result and write down your reasoning clearly.

Solution

- EMP.dno is the foreign key to DEPT. So the join size is equal to the size of EMP = 30000 records.
- After selection 2% of EMP tuples are obtained = 30000x2% = 600 records.
- EMP.name values are almost distinct. After projection the result = 600 records.

Question 3. Consider a database consisting of the following three relation schemas. The meaning of the attributes in the above schemas is self-explanatory. For example, Bname is the name of the branch. Keys are underlined and foreign keys are in italics.

- CUSTOMERS (CID, Cname, job, age, street, city): 10,000 tuples
- BRANCHES (BID, Bname, city): 2,000 tuples
- ACCOUNTS (CID, BID, balance): 12,000 tuples

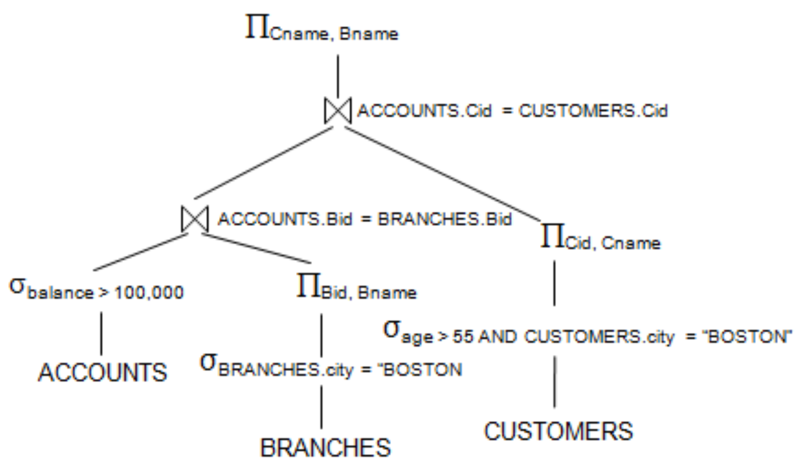
Consider the following SQL query:

```
SELECT    Cname, Bname, balance

FROM      CUSTOMERS, BRANCHES, ACCOUNTS

WHERE     CUSTOMERS.cid = ACCOUNTS.cid

AND       BRANCHES.bid = ACCOUNTS.bid

AND       age > 55

AND       CUSTOMERS.city = "BOSTON"

AND       BRANCHES.city = "BOSTON"

AND       balance > 100,000
```

Assume that 30% of customers are older than 55, 20% of customers live in BOSTON, 10% of branches are located in BOSTON and 30% of balance are greater than 100,000.

Using the above information and the heuristic rules for optimizing a query that you have learned in the course, draw the most efficient query tree for the SQL query. You should clearly indicate all the nodes and branches in the tree. Justify your answer by calculating the size of intermediate results.

Solution:



Justification:

The selectivity of the operation age > 55 on CUSTOMERS is 30%.

The selectivity of the operation CUSTOMERS.city = “BOSTON” on CUSTOMERS is 20%.

The selectivity of the operation BRANCHES.city = “BOSTON” on BRANCHES is 10%.

The selectivity of the operation age > 55 AND city = “BOSTON” on CUSTOMERS is 30%*20% = 6%.

The selectivity of the operation balance > 100,000 on ACCOUNTS is 30%.

The estimated size of σage > 55 AND CUSTOMERS.city = “BOSTON” CUSTOMERS is 6% * 10, 000 = 600 tuples.

The estimated size of σBRANCHES.city = “BOSTON” BRANCHES is 10% * 2,000 = 200 tuples.

The estimated size of σbalance > 100,000 ACCOUNTS is 30% * 12,000 = 3,600 tuples.

The estimated size of CUSTOMERS⋈ACCOUNTS is (600 x 3600)/600 = 3600 tuples.

The estimated size of BRANCHES⋈ACCOUNTS is (3,600x 200)/200 = 3600 tuples.

The estimated size of CUSTOMERS⋈BRANCHES = 600 * 200 = 120,000 tuples.

The intermediate result should be as small as possible, so we join BRANCHES and ACCOUNTS first and then join the intermediate result with CUSTOMERS. Another option is join CUSTOMERS and ACCOUNTS first and then join the intermediate result with BRANCHES. This is the trickiest part of the question: join which two relations out of the three first.