National University of Computer and Emerging Sciences
Lahore Campus

Advance Database Concepts (CS4064)

Assignment 3
Total Marks: 150

TA: Muhammad Zayam Amjad

Hasan Yahya
Name

22L-7971
Roll No.

BSE-6A
Section

Do not write below this line.

Note: Please ensure that you attempt all questions and their respective parts in the given order.

Q. No 1: Consider the following part of library database schema and the query in *SQL* and *RA*:

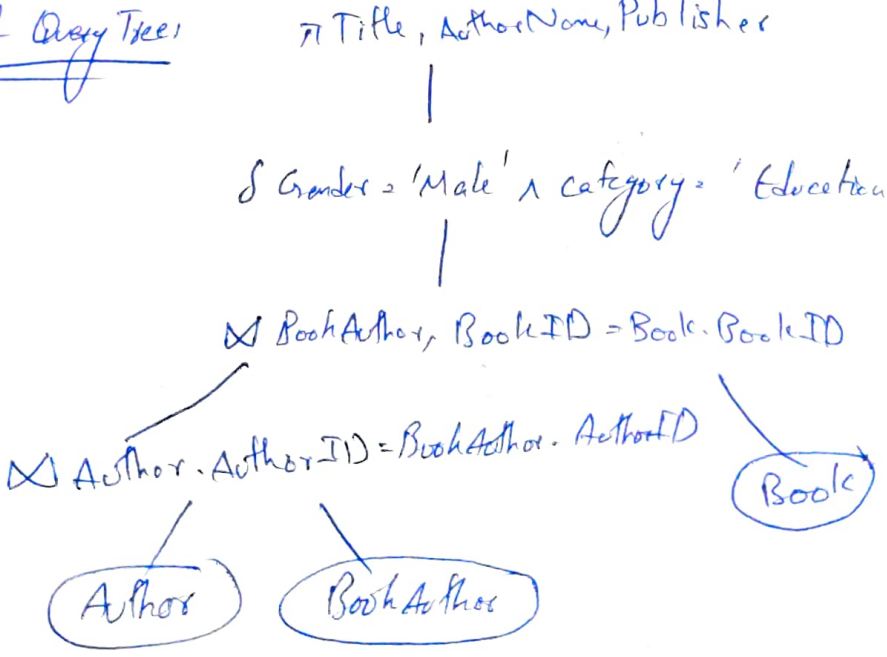
Book (BookID, Title, Category, Publisher, PublishYear)
Author (AuthorID, AuthorName, Gender, Email, OriginCity)
BookAuthor (BookID, AuthorID)

```
SELECT Title, AuthorName, Publisher
FROM Author A JOIN BookAuthor BA ON A.AuthorID=BA.AuthorID JOIN Book B ON B.BookID=BA.BookID
WHERE Gender= 'Male' AND Category= 'Education';
```

$\pi_{Title, AuthorName, Publisher}(\sigma_{Gender='Male' \wedge Category='Education'}(Author * BookAuthor * Book))$

Your task is to optimize this query and draw the best possible query tree for this query. Take appropriate database statistics to support your answer. (Show all steps) [10]

Initial Query Tree:



Heuristics

push down

1. Apply select early
2. Apply project early
3. Combine joins
4. Replace cartesian Product with join

Assuming:

- Author = 10000 rows
  - Gender = male 50% of them → 5000 rows
- Book = 50000 rows
  - category 'Education' is 10% of them so 5000 rows
- BookAuthor = 100,000 rows

Applying Heuristics to Optimize

1. Apply selection early:

π Title, AuthorName, Publisher

$\bowtie$ BookAuthor.BookID = Book.BookID (10000 tuples)

$\bowtie$ Author.AuthorID = BookAuthor.AuthorID (50000 tuples)

σ Gender = 'Male' (5000 tuples)

Author

BookAuthor (100000 tuples)

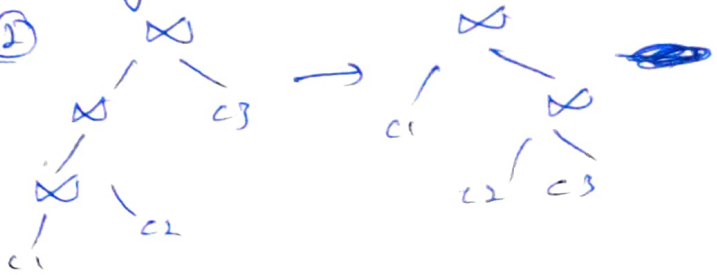
σ category = 'Education'

Book

→ Final output gives us 25000 tuples

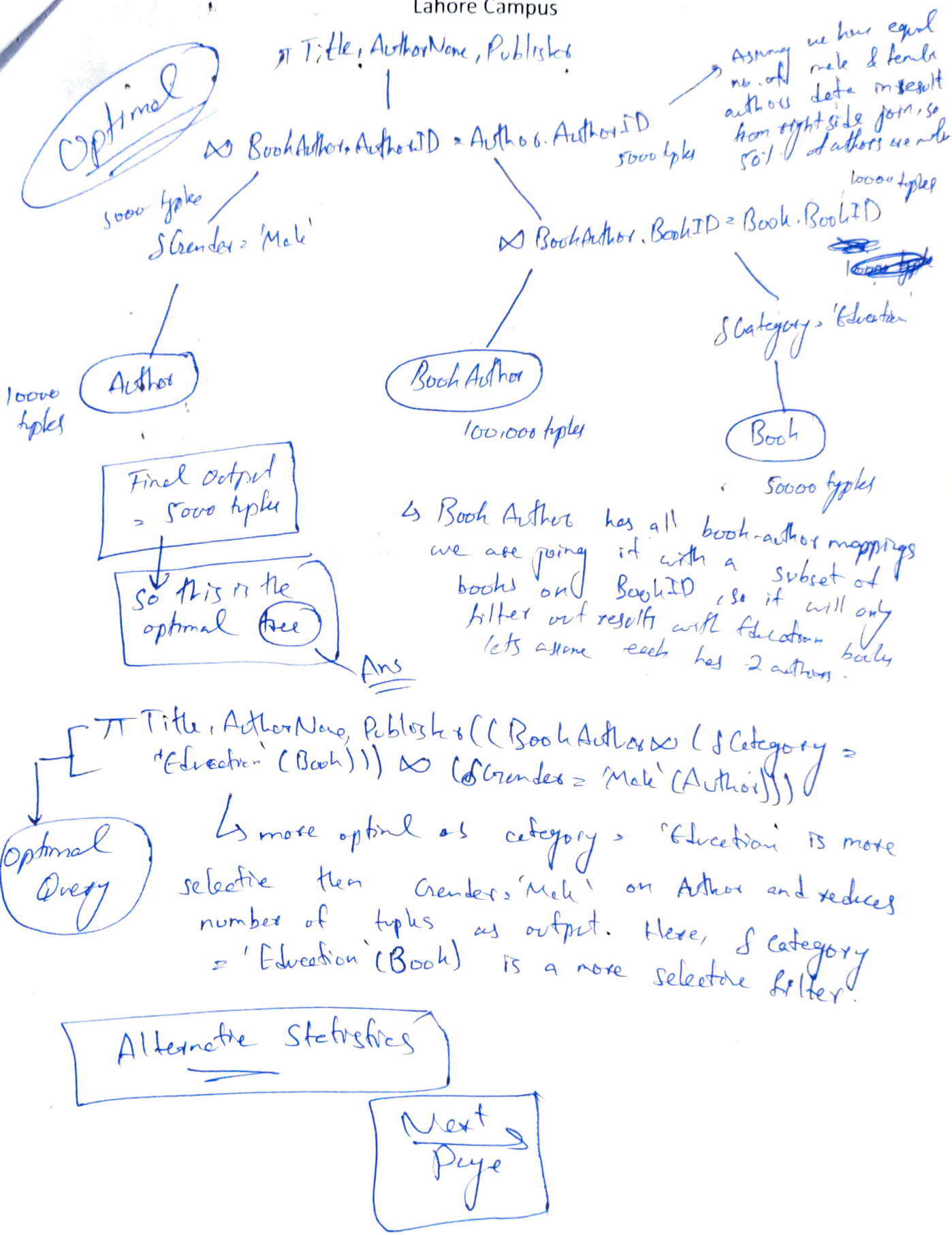
→ If we apply rules of equivalence to find possible query trees:

Rule (2)



Rule (3) of equivalences applies as well

→ [σ Gender = 'Male' (Author)
σ category = 'Education' (Book)]


Optimal
π Title, AuthorName, Publisher
⋈ BookAuthor.AuthorID = Author.AuthorID
5000 tples
Assuming we have equal no. of male & female authors data in result from right side join, so 50% of authors are male
10000 tples
5000 tples
σ Gender = 'Male'
⋈ BookAuthor.BookID = Book.BookID
σ Category = 'Education'
10000 tples
Author
BookAuthor
Book
100,000 tples
50000 tples
Final output = 5000 tples
So this is the optimal tree
Ans
↳ BookAuthor has all book-author mappings we are going it with a subset of books on BookID, so it will only filter out result with Education books lets assume each has 2 authors.
π Title, AuthorName, Publisher((BookAuthor ⋈ (σ Category = 'Education' (Book))) ⋈ (σ Gender = 'Male' (Author)))
Optimal Query
↳ more optimal as category = 'Education' is more selective then Gender = 'Male' on Author and reduces number of tples as output. Here, σ Category = 'Education' (Book) is a more selective filter.
Alternative Strategies
Next Page

Tables	Rows	Selectivity (Filter)	Estimate Result
Author	10000	Gender = 'Male' (50%)	5000
Book	200000	Category = 'Education' (10%)	20000
BookAuthor	40,000	N/A	40000

→ 5000 × 40000 → small result join will give 2000 rows instead of All (20000)

(If you went to show projections at each split you can also write the same query tree as Copy of 1)

π Title, Authorname, Publisher

⋈ BookAuthor.AuthorID = Author.AuthorID

π Title, AuthorID, Publisher

⋈ BookAuthor.Book = Book.BookID

π BookID, Title, Publisher

σ Category = 'Education'

Book

Book Author

π Gender, AuthorID, AuthorName

σ Gender = 'Male'

Author

(Same with extra projection specified)

CLO # 3: To develop a solution for given scenario/challenging problem in the domain of DB systems.

Q. No 2: [5+5]

a. Consider the above library database schema and the query in *SQL/RA* given in *Q#1*. Assume that the frequency of access of this query is very high. Which attributes are more appropriate to create indexes for an efficient execution strategy to improve the performance of this query?

We want to index columns that are either in the WHERE clause or JOIN clause/conditions (to match rows efficiently). So we have 3 suggested attributes

→ Author.Gender (used in WHERE clause)

→ Book.Category (used in WHERE clause)

→ Joining columns for all 3 tables (ie Author.AuthorID, BookAuthor.AuthorID, Book.BookID, BookAuthor.BookID).

b. Consider the above library database schema and assume that *Book*, *Author* and *BookAuthor* tables have *100000*, *100000* and *50000* rows respectively. Estimate the potential *Join Cardinality* (*jc*) of *Book* $\bowtie_{BookID=BookID}$ *BookAuthor* (i.e., max number of rows resulting from the inner join of these two tables). Justify your answer.

Since there is an innerjoin being used and BookID is key for Book table, joining will yield at most 50000 tuples. (ie min of the 3).

As we know, join selectivity is,

$$js = 1 / MAX(NDV(bookID, Book), NDV(bookID, bookAuthor))$$

$$= 1 / 100000 = \boxed{0.000001} \text{ — Ans}$$

and join cardinality is,

$$jc = js \times |Book| \times |BookAuthor|$$

$$jc = (0.00001) \times 100000 \times 50000$$

$$\boxed{jc = 50000} \text{ — Ans}$$

CLO # 3: To develop a solution for given scenario/challenging problem in the domain of DB systems.

Q. No 3: Assume: A block size is *B= 1024* bytes, file has *r= 1,000,000* records, each record is *100* bytes long, a block pointer is *P= 10* bytes, a record pointer is P_R= *11* bytes, and a key field for the index is 6 bytes long. A database system uses a B+-trees index on a key field. A leaf node and non-leaf node are one block in size and contain as many keys (and appropriate pointers) as will fit in a block. How many blocks will this index use? Also estimate the number of block accesses needed to search for and retrieve a record from the file given its key value using the B+-tree index. Show your working. [10]

(4) Order of P:

$$(p \times P) + ((p-1) \times V_{SSN}) < B$$

$$(p \times 10) + ((p-1) \times 6) < 1024$$

$$10p + 6p - 6 < 1024$$

$$16p < 1030$$

$$p < 1030/16 \approx 64.375$$

So, $p = 64$

(4) Order of P_{leaf}:

$$(P_{leaf} \times (V_{SSN} + P_R)) + P < B$$

$$(p_{leaf} \times (6+11)) + 10 < 1024$$

$$17\, p_{leaf} < 1024 - 10$$

$$17\, p_{leaf} < 1014$$

$$p_{leaf} < 1014/17$$

$$p_{leaf} < 59.64$$

$p_{leaf} = 59$

At level-01 (b1):

$$b1 = \left\lceil \frac{r}{pleaf} \right\rceil = \left\lceil \frac{1000000}{59} \right\rceil = 16950 \text{ blocks}$$

At level-02 (b2):

$$b2 = \left\lceil \frac{16950}{64} \right\rceil = 265 \text{ blocks}$$

At level-03 (b3):

$$b3 = \left\lceil \frac{265}{64} \right\rceil = \lceil 4.14 \rceil = 5 \text{ blocks}$$

At level-04 (b4):

$$b4 = \left\lceil \frac{5}{64} \right\rceil = 1 \text{ block}$$

Total block accesses = 16950 + 265 + 5 + 1

= **17221 blocks**

Ans

Block access cost to search a key value = x + 1 = 4 + 1 = **5 blocks**

Ans

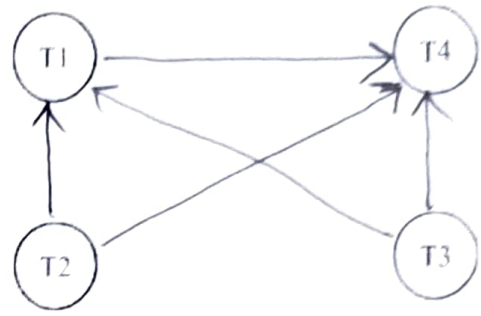
CLO # 2: Apply the models and approaches to become enabled to select and apply appropriate methods for a particular case.

Q. No 4: Consider the following schedule: [5]

S: r1(X), r2(Z), w1(Z), r3(X), r3(Y), w1(X), w3(Y), r4(Y), w4(Z), w4(Y).

Draw the serializability (precedence) graph for this schedule. State whether this schedule is conflict-serializable (correct) or not. If the schedule is conflict-serializable, write down the equivalent serial schedule(s) otherwise explain why it is not. *Also state whether this schedule is view-serializable or not.*

T1	T2	T3	T4
R(X)			
	R(Z)		
W(Z)		R(X)	
		R(Y)	
W(X)			
		W(Y)	
			R(Y)
			W(Z)
			W(Y)



Since there is no loop in the graph, it is conflict-serializable with equivalent serial schedules as T2 → T3 → T1 → T4 and T3 → T2 → T1 → T4.

Since, all conflict serializable schedules are also view serializable, this graph is also view serializable.

CLO # 2: Apply the models and approaches to become enabled to select and apply appropriate methods for a particular case.

Q. No 5: Consider the following schedule of actions: [10+10]

S: r1(X), r2(Z), w1(Z), r3(X), r3(Y), w1(X), w3(Y), r2(Y), c3, c2, c1, w4(Z), w4(Y), c4.

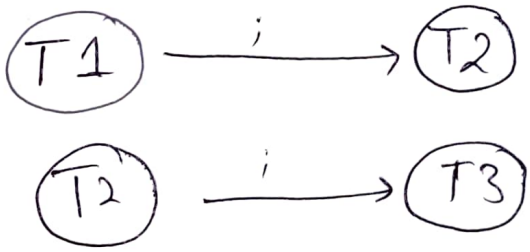
For each of the following concurrency control mechanisms, describe how the concurrency control mechanism handles the schedule. Assume that the timestamp of transaction *Ti* is *i*. For lock-based concurrency control mechanisms, add lock and unlock requests to the above schedule of actions as per the locking protocol. The DBMS processes actions in the order shown. If a transaction is blocked, assume that all its actions are queued until it is resumed; the DBMS continues with the next action (according to the listed schedule) of an unblocked transaction.

a. Rigorous 2PL with timestamps used for deadlock detection (Use wait-for-graph to deal with deadlock)
b. Basic Timestamp Ordering (Assume T1 < T2 < T3)

a) → [S (ie S1, S2, ... are for shared locks)
X (ie X1, X2, ... are for exclusive locks).

T_1	T_2	T_3	T_4
S1_lock(X) r1(X)			
X1_lock(2) wait for T2.	S2_lock(2) r2(2)		
		S3_lock(X) r3(X) S3_lock(Y) r3(Y) X3_lock(Y) upgrade-lock w3(Y)	
	S2_lock(Y) wait for T3	c3 – release locks unlock3(X) unlock3(Y)	
	r2(Y) → wakeup restart c2 – release locks unlock1(2) unlock2(Y)		
w1(2) – wakeup X1_lock(X) – upgrade w1(X) c1 – release unlock(2) unlock(X)			X4_lock(2) w4(2) X4_lock(Y) w4(Y) c4 – release locks all ~~unlock(2)~~ unlock 4(2) unlock 4(Y)

Wait-for-graph:



not enough space so i put some here.

b)

T1	T2	T3	T4
R1(X)			
	R2(Z)		
W1(Z) → abort T1 as RTS(Z) > TS(T1)			
		R3(X)	
		R3(Y)	
		W3(Y)	
	R2(Y) – abort WTS(Y) > TS(T2)		
		C3	
			W4(Z)
			W4(Y)
			C4
with new timestamp ← restart T1			
	Restart T2 with new timestamp		

X		Y		Z	
RTS	WTS	RTS	WTS	RTS	WTS
TO	{ }	TO	{ }	TO	{ }
{T1}					
				{T2}	
{T3}					
		{T3}			
			T3		
					T4
			T4		

imp

Note, the dots are drawn to show which level each entry is in

CLO # 2: Apply the models and approaches to become enabled to select and apply appropriate methods for a particular case.

Q. No 6: Consider the following log at the point of system crash. Suppose that we use ARIES recovery algorithm to answer the following questions. [9]

LSN	Last_LSN	Trans_ID	Type	Page_ID	Other_Info
1	0	T1	Update	A	…
2	0	T2	Update	C	…
3	1	T1	Commit		…
4	2	T2	Update	A	…
5	begin checkpoint				
6	end checkpoint				
7	4	T2	Commit		…
8	0	T3	Update	B	…
9	0	T4	Update	C	…
10	8	T3	Update	A	…
11	9	T4	Commit		…

Master Log: LSN = 5

a. Show the contents of transaction and dirty page table at the time of checkpoint. [2]

Trans_ID	LSN	Status
T1	~~1~~ 3	commit
T2	~~2~~ 4	in progress
—	—	—

Page_ID	LSN
A	1
C	2
—	—

b. What is done during Analysis? Be precise about the points at which Analysis begins and ends and show the contents of transaction and dirty page table reconstructed in this phase. [3]

5 (i.e. begin checkpoint)

Analysis Phase: [redacted] **From:** LSN = [redacted] **To:** LSN = 11

Transaction Table

Trans_ID	LSN	Status
T1	3	commit
T2	7	commit
T3	10	in progress
T4	11	commit

Page_ID	LSN
A	[redacted] 1
[redacted] B	[redacted] 8
[redacted] C	[redacted] 2

Dirty Page Table

Analysis phase starts from last checkpoint and updates the values of Transaction table & dirty page table which in turn will later be used in the coming UNDO and REDO phases.

c. What is done during Redo? Be precise about the points at which Redo begins and ends. [2]

Begin_LSN = 1 END_LSN = 10

LSNs (1,2,4,8,9,10) update the corresponding pages (A,C,A,B,C,A).

→ Redo does all operations from start to end except for uncommitted ones.

d. What is done during Undo? Be precise about the points at which Undo begins and ends. [2]

Begin_LSN = 10 End_LSN = 8

LSNs (10,8) and corresponding pages (A,B), undo will only perform on T3.

→ Undo starts from end & rollbacks uncommitted/incomplete transactions.

Q. No 7: Consider the bank database, and the following SQL query: [5+5+5]

Customer (custID, custName, cnic, birthDate, address, ...)

Account(accNo, custID, accTitle, accType, openingDate, ...)

Transaction(tID, accNo, transType, amount, transDate, ...)

no of Trans

Applying Heuristics because it result already optimized.

Write an efficient relational-algebra expression that is equivalent to these queries and draw the optimal query plan for this query.

To RA: π C.cnic, A.accNo, A.title, T.no of Trans (Customer $\bowtie$ (Account $\bowtie$ (T.accNo (γ accNo, count(*) (Transaction)))).

1. **SELECT** C.cnic, A.accNo, A.Title, T.noOfTrans **FROM** customer C **JOIN** account A **ON** C.custID=A.custID **JOIN** (SELECT accNo, COUNT(*) **AS** noOfTrans FROM transaction **GROUP BY** accNo) T **ON** A.accNo=T.accNo

RA Expression: (optimized)

π Cnic, accTitle, no of Trans, accNo (

(π CustID, Cnic (Customer)) $\bowtie$ CustID = CustID

((π accNo, custID, accTitle (Account))

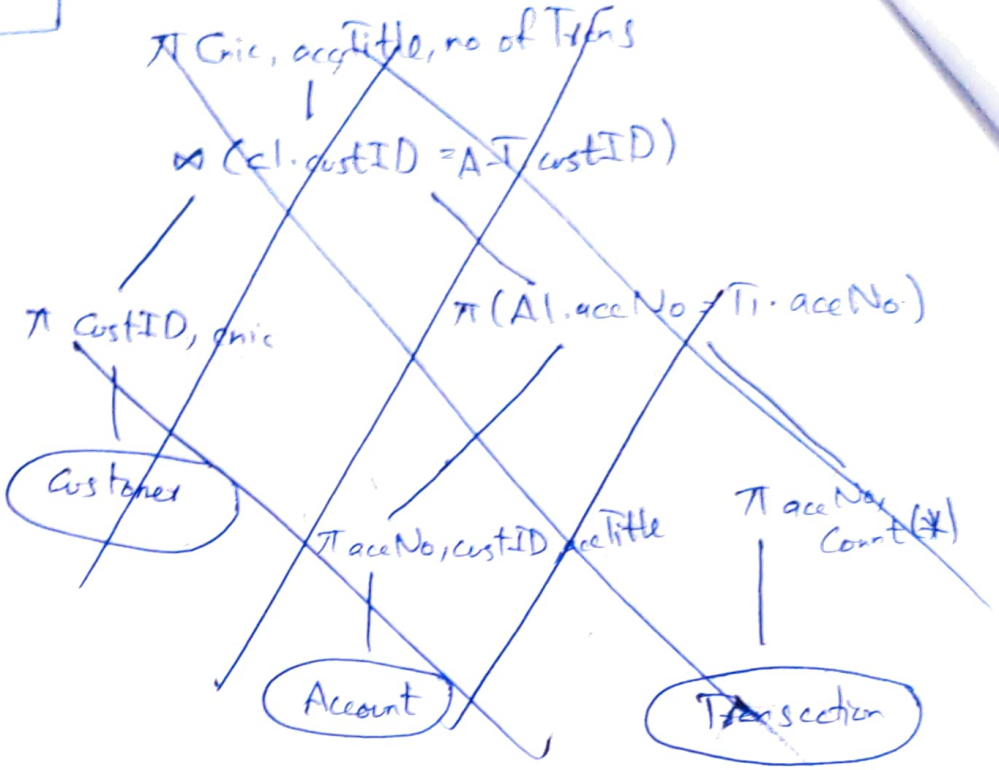
$\bowtie$ accNo = accNo (γ accNo: Count(*) noTrans (Transactions)

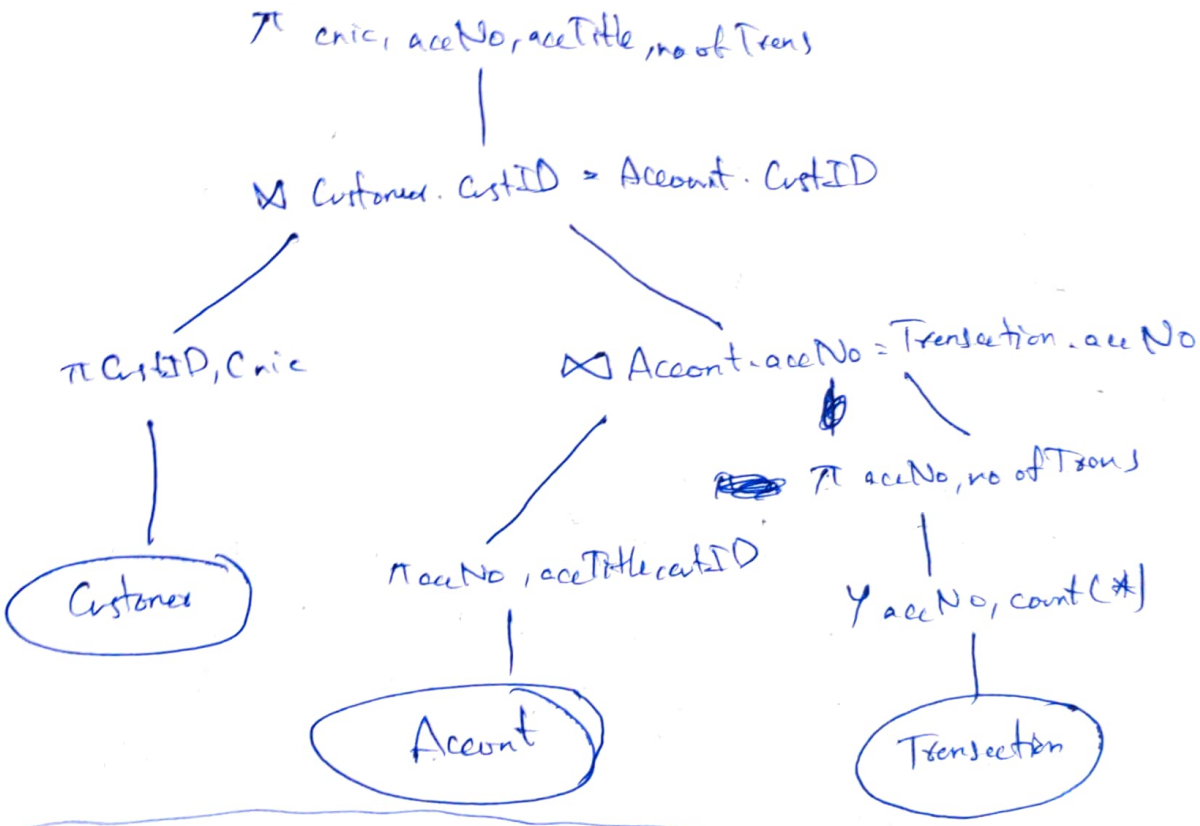
))

)

Next → Page

Optimized Query Trees





The original Query's RA tree

→ Account ⋈ (T.accNo(γ accNo, count(*) Transaction)) T.accNo, count(*)
(γ accNo count(*) (Transaction)).

→ π c.cnic, A.accNo, A.Title, T.no of Trans (Customer ⋈ Account ⋈
(T.accNo (γ accNo, count(*) (Transaction)))).

we apply Heuristics to [redacted] to get the optimal query & tree.

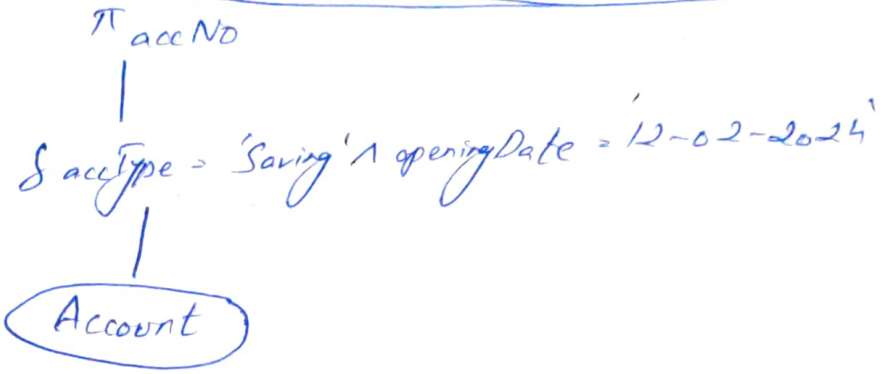
2. **SELECT** accNo **FROM** Account **WHERE** accType='Saving' **AND** openingDate='12-02-2024'

RA Expression: (optimized)

π accNo (σ accType = 'Saving' $\wedge$ openingDate = '12-02-2024' (Account))

Optimized Query Tree:

This query is already efficient as it applies selection first and then projection to reduce no of tuples



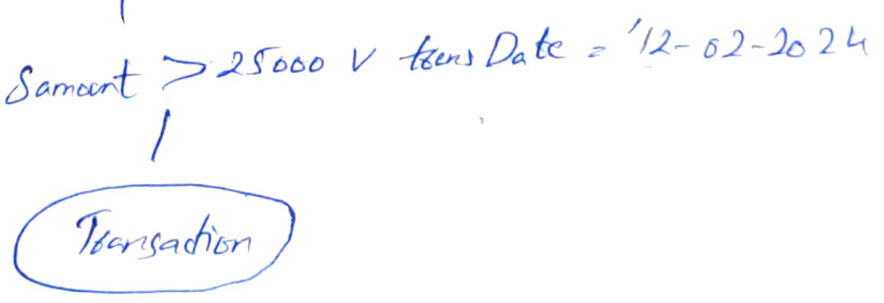
3. **SELECT** accNo **FROM** Transaction **WHERE** amount >25000 **OR** transDate ='12-02-2024'

RA Expression: (optimized)

π accNo (σ amount > 25000 $\vee$ transDate = '12-02-2024' (Transaction))

Optimized Query Tree:

Already efficient as selection is applied early then projection to reduce no of tuples



→ **Alternative for (2)** we could trim the columns first & then selection & then projection.

i.e.,

Query:

$$\pi_{accNo}(\sigma_{accType = 'Saving' \wedge openingDate = '12-01-2024'}(\pi_{accType, openingDate}(Account)))$$

This approach reduces the tuple width early on & keeps only necessary attributes in the memory

But this strategy is only effective if Account has a lot of columns (10-20) around. The ('---') in the Bank database shows tables may have more columns

Optimized Query Tree:

~~$\pi_{accNo}(\sigma_{accType = 'Saving' \wedge openingDate = '12-01-2024'}$~~

~~$(\pi_{accNo, accType}$~~

π_{accNo}

|

$\sigma_{accType = 'Saving' \wedge openingDate = '12-01-2024'}$

|

$\pi_{accNo, accType, openingDate}$

|

Account

Q. No. 8: Assume that the number of buffers available in main memory for implementing the join is k= n_b = 5 blocks. The DEPARTMENT file consist of r_D = 40 record stored in b_D = 10 disk blocks and that the EMPLOYEE file consists of r_E = 600 record stored in b_E = 200 disk blocks. [3*5=15]

Suppose that secondary indexes exist on both the attributes
Ssn of employee and *Mgr_ssn* of department,
with the number of index levels X_{Ssn} = 4 and X_{Mgr_ssn} = 2, respectively.

Write cost = $((js \times |R| + |S|)/bfrRS)$

DEPARTMENT $\bowtie_{Mgr\_ssn=Ssn}$ **EMPLOYEE**

Apply below Joining Algorithms and find cost estimation. Suggest what will be the best joining algorithm to apply on above case.

1. **Nested-loop join (nested-block join)**

if departmnt records in outer loop,

$C_{j1} = b_D + (\lceil b_D/(n_B-2)\rceil \times b_E) + ((js \times |R| \times |S|)/bfrRS)$ ← write cost

~~$10 + (\lceil 10/(5-2)\rceil \times 200) + ((js \times 40 \times 600)/bfrRS)$~~

$C_{j1} = 10 + (\lceil 10/(5-2)\rceil \times 200)$ + write cost

$= 10 + (4 \times 200) = 10 + 800 = 810$ + Write Cost

or if we employee records in outer loop,

$C_{j1} = 200 + (\lceil 200/3\rceil \times 10)$

$= 870$ + write cost

Ans = 870 + write cost departmnt in outerloop

so, using is better.

2. **Index-based nested-loop join**

if departmnt in outer loop,

$C_{j2a} = b_D + |R| \times (x_{Mgr_ssn} + 1 + s_{Mgr_ssn}) + $ write cost

$= 10 + 40 \times (4 + 1 + 1)$

$= 10 + (40 \times 6) = 250$ + write cost

~~if employee records in outer loop;~~

~~$= 200 + 600 \times (2 + 1 + \frac{40}{600})$~~

Next → Page

if Employee records in outer loop,

$= bE + (rE \times (X_{mng\text{-}SSN} + 1)) + \text{write cost}$

$= 200 + (600 \times (2+1)) + \text{write cost}$

$= 200 + (600 \times 3) + \text{write cost}$

$= 200 + (1800) + \text{write cost}$

$= \boxed{2000} + \text{write cost}$

Ans

if Department records in outer loop,

$= bD + (rD \times (X_{SSN} + 1)) + \text{write cost}$

$= 10 + (40 \times (4+1)) + \text{write cost}$

$= 10 + (40 \times 5) + \text{write cost}$

$= \boxed{210} + \text{write cost}$

Ans

So Department records in outer loop is better.

3. **Sort-merge join**

→ if both E and D are already sorted,

Cost = $b_E + b_D$ + write cost

= 200 + 10 = **210** — Ans + write cost

→ if they are not sorted,

Cost = $b_E + b_D + b_E \log_2 b_E + b_D \log_2 b_D$ + write cost

= $200 + 10 + 200 \log_2(100) + 10 \log_2(10)$ + write cost

= **1771.9905** — Ans + write cost

4. **Partition-hash join**

Cost = $3 \times (b_E + b_D)$ + write cost

= $3 \times (200 + 10)$ + write cost

= $3 \times (210)$ + write cost

= **630** — Ans + write cost

Conclusion:

→ If we can control if data is sorted or unsorted or which attribute should be in outer loop then, it is better to do index based nested loop joins with Department records in outer loop. (cost = 210). However, if we don't have control over them then, the worst-case cost of partition hash (630) is better than worst case cost of another.

→ **Hash joins** are almost always better than sorting based join algorithms, but there are cases where sorting based joins are

preferred, (such as on non-uniform data or where data is sorted on join keys or where result itself needs to be sorted).

Condition	Sub-condition	Cost
Nested Loop (Block)	Outer = Employee	870
	Outer = Department	810
Nested Loop (Index)	Outer = Employee	2000
	Outer = Department	210
Sort Merge	Sorted	210
	Unsorted	1771.965 = 1772
Partition Hash	✗	630

Lowest

Q. No. 9: Write down the Cost function of the following selection operations [1*5=5]

1. Primary index to retrieve a single record (3a)

$$C_{s3a} = x + 1$$

2. A hash key to retrieve a single record (3b)

$$C_{s3b} = 1$$

3. An ordering index to retrieve multiple records (4)

$$C_{s4} = x + \frac{b}{2}$$

4. A clustering index to retrieve multiple records (5)

$$C_{s5} = x + \lceil s / bfr \rceil$$

5. Secondary(B+-tree) index in worst case as well as for range queries (6a & 6b)

$$C_{s6a} = x + 1 + s \quad \text{(worst case)}$$

$$C_{s6b} = x + \frac{b_m}{2} + \frac{r}{2} \quad \text{(for large queries)}$$

Now consider following Statistics of EMPLOYEE

→ Assuming [4*8=32] this is 200.

Suppose that the EMPLOYEE file has r_E = 1000 records stored in b_E = 2,00 disk blocks with blocking factor bfr_E = 5 records/block and the following access paths:

1. A clustering index on Salary, with levels x_{Salary} = 3 and average selection cardinality s_{Salary} = 20.

   Find a selectivity of sl_{Salary} ?

$$sl_{salary} = \frac{20}{1000} = \boxed{0.02} \quad \text{Ans}$$

2. A secondary index on the key attribute Ssn, with xSsn = 4 and average selection cardinality s_{Ssn} = 1, what would be the $sl_{Ssn=?}$

$SI_{SSN} = \frac{1}{1000} = 0.001$ — Ans

$= \frac{S_{SSN}}{rE}$

3. A secondary index on the nonkey attribute Dno, with x_{Dno} = 2 and first-level index blocks b_{I1Dno} = 4. There are *NDV (Dno, EMPLOYEE)* = 125 distinct values for Dno, so what would be the the selectivity of Dno is sl_{Dno} = ? and the selection cardinalty is s_{Dno} = ?

$SL_{DNO} = (1/NDV(Dno, Employee)) = \frac{1}{125} = 0.008$

Ans

$S_{DNO} = (rE \times SL_{DNO}) = (rE / NDV(Dno, Employee))$

$= (1000 \times 0.008) = 8$ — Ans

4. A secondary index on Sex, with x_{Sex} = 1. There are *NDV (Sex, EMPLOYEE)* = 2 values for the Sex attribute, so the average selection cardinality is s_{Sex} =?

$S_{sex} = (rE / NDV(Sex, Employee))$

So,

$S_{sex} = (1000 / 2) = 500$ — Ans

The use of cost functions with the following examples and suggest the best cost function for each operation.

OP1: σSsn='123456789' (EMPLOYEE)

On this we can either use Linear Search (S1) or Secondary B+ index (S6a) for worst case

$C_{S1b} = \left(\frac{b_E}{2}\right) = \frac{200}{2} = \boxed{100}$

Best ← $(C_{S6a} =)$ $x_{SSN} + 1 = 4 + 1 = \boxed{5}$ — best cost function is B+ index (worst case) ie $C_S = x_{SSN} + 1$.

OP2: σDno > 5 (EMPLOYEE)

On this we can either use Linear Search (S1) or Secondary B+ index for range (S6b),

Best ← $(C_{S1a} =)$ $b_E = \boxed{200}$ — operation on non-key attribute.

$C_{S6b} = x_{Dno} + (b_{I1Dno}/2) + (r_E/2)$
$= 2 + (4/2) + (1000/2) = \boxed{504}$
Best to use linear search on non-key.

OP3: σDno = 5 (EMPLOYEE)

On this we can either use method S1 or S6a, ie (Linear search vs B+ index for worst case)

$C_{S1a} = b_E = \boxed{200}$

Best ← $(C_{S6a} =)$ $x_{Dno} + s_{Dno} = 2 + 8 = \boxed{10}$ Best to use S6a ie B+ index for worst case

OP4: σDno=5 AND SALARY>30000 AND Sex='F' (EMPLOYEE)

We can use either Linear approach, or we can use anyone of three components of selection condition so, we need to check all 4 of them.

→ Linear Approach (S1):

$C_{S1a} = b_E = \boxed{200}$ — non key ~~this is the best cost function~~

→ Dno indexed access path (S6a):

$C_{S6a} = x_{Dno} + s_{Dno} = 2 + 8 = \boxed{10}$ → Best

↳ using B+ worst case

Next Page →

→ cost of salary indexed access path (S5):

clustering index

$C_{S5} = X_{salary} + S/bfr = 3 + \frac{500}{5}$

↳ Here 500 assumes 1000/2 ie half employees qualify.

$= \boxed{103}$

→ cost of Sex indexed access path (S6a)

B+ worst case

$C_{S6a} = X_{sex} + S_{sex} = 1 + 500 = \boxed{501}$

Best is S6a B+ worst case

on Dno — Any (ie S6a)

if student in outer loop

bS + (⌈bS/nb-2⌉ × bD) = 2000 + (⌈2000/1⌉) × 10 = 22000

but since this is 22000 > 2010 blocks we don't use it & use the other one. ie min

Q. No. 10: Assume that the number of buffers available in main memory for implementing the join is **no = 3 blocks**. The DEPARTMENT file consists of **r(D) = 50 records** stored in **b(D) = 10** disk blocks and the Student file consists of **r(S)= 6000 records** stored in **b(S) = 2000 disk blocks.** **[1+(3*6)=19]**

a) How many total number blocked fetched by using Nested loop join strategy if we performed following query [1]

department in outer loop

Select * from Department D join Student S ON D.DID=S.DID

Total Blocks= bD + (⌈bD/nb-2⌉ × bS) = 10 + (⌈10/1⌉ × 2000)

= 10 + 20000 = 2010 blocks

Ans

b) **What would be Sort-merge join cost if:**

1. If both student and Department are already sorted

Cost = bD + bS = 10 + 2000 = 2010 — Ans

2. If both are not sorted and sorted cost using external sort-merge algorithm is (b loga b) b is the total number of blocks in a file

Cost = bD + bS + bD $\log_2$ bD + bS $\log_2$ bS

= 10 + 2000 + 10 $\log_2$(10) + 2000 $\log_2$(2000)

= 23974.787 — Ans

c) Suppose the Student table has 80 distinct values over the attribute DID and the Department has 50 distinct values over the attributa DID.

Select * from Department D join Student S ON D.DID=S.DID

1. Find join selectivity ratio of both tables

Js= $\frac{1}{\max(V_D, V_S)}$ = $\frac{1}{\max(80, 50)}$ = $\frac{1}{80}$ = 0.0125

Ans

2. Find join cardinality of both tables

Jc= r(D) * r(S) * Js = 50 * 6000 * 0.0125

= 3750 records — Ans

Select * from Department D where D.DID NOT IN (Select S.DID from Student S) → Anti-join

3. Find join selectivity ratio of both tables

Js= $1 - MIN(1, NDV(T2.y)/NOV(T1.x))$

$= 1 - MIN(1, 80/50)$

$= 1 - MIN(1, 1.6) = 1 - 1 = \boxed{0}$ — Ans

4. Find join cardinality of both tables

Jc= $|T_1| * js = 50 * 0 = \boxed{0 \text{ records}}$ — Ans

:: < ~ Good luck ~ > ::