University Echahid Hamma Lakhdar, El Oued

Department of Computer Science | Academic Year: 2025/2026

1st Y Master IoT & AI | Duration: 1h30

Final Exam of Advanced Databases

Exercise 01 (3 marks)

A block nested loop join is executed with S as the outer relation and R as the inner relation. The join completes in 2 iterations. In each iteration, a block of 3 pages of S are loaded into the buffer. For each block of S, 10 pages of R are loaded into the buffer one by one

1) How many pages does S have, and how many pages does R have? Justify your answer.

2) Compute the total number of pages I/Os based on the simulation steps.

1) S = 6 pages, R = 10 pages

2) Iteration 1: load 3 pages of S → 3 I/Os for S
For each S block, load R (10 pages) → 10 I/Os for R
Iteration 2: load next 3 pages of S → 3 I/Os for S
For each S block, load R → 10 I/Os for R

Total 6+2*10 = 26 IOs

Exercise 02 (3 marks)

Given the Sort-Merge Join algorithm, where during sorting phase we observed:

- After the initial sort pass of R, the system produces 2 sorted runs of 5 pages each. Then, these 2 runs are merged in one merge pass
- After the initial sort pass of S, the system produces 2 sorted runs of 5 and 1 pages. Then, these 2 runs are merged in one merge pass

1) Deduce the buffer size and number of merge passes for each relation during the sorting phase.

2) Compute the total number of pages I/Os for R⋈S using Sort-Merge Join algorithm.

1) Buffer size (B)

In external sorting, the initial run size = B pages (all fit in memory).

- For R, initial runs are 5 pages ⇒ B = 5 pages
- For S, one run is 5 pages (and the remainder is 1 page) ⇒ consistent with B = 5 pages

So Buffer size = 5 pages

Number of merge passes

R: 2 initial runs → merged into 1 run in 1 merge pass

S: 2 initial runs → merged into 1 run in 1 merge pass

2) Total page I/Os for R⋈SR \bowtie SR⋈S using Sort–Merge Join

- **Total for R sorting** read 10 + write 10 + read 10 + write 10 **=** 40 I/Os
  Initial sort pass Merge pass
- **Total for S sorting=** read 6 + write 6 + read 6 + write 6 = 24 I/Os
  Initial sort pass Merge pass

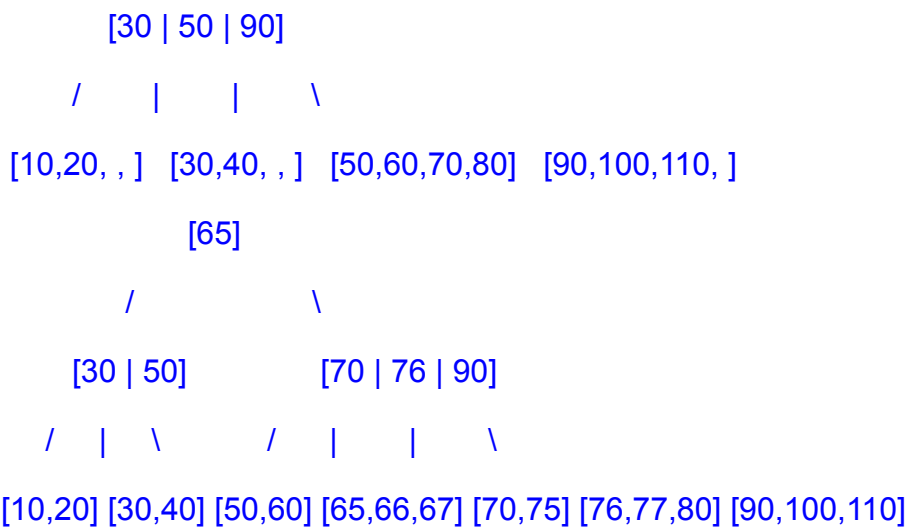
Total Sort Merge Join cost = 40 (sort R)+24 (sort S)+10+6 (join)=80 page I/Os

Exercise 03 (4 marks)

Consider a B+ Tree of order d=3 used in the course (root and intermediate nodes: 5 pointers and 4 keys. Leaf node: 4 entries). The B+ Tree is initially empty.

1) Show the resulting tree after inserting the following keys into the B+ Tree in the sequence:
50, 30, 90, 100, 20, 10, 40, 60, 70, 80, 110

2) Now, starting from the tree obtained in Question 1, insert the following keys: 65, 66, 67, 75, 76, 77



Exercise 04 (10 marks)

Consider the following relational schema:

Suppliers(*sid:* `integer`, *sname:* `char(20)`, *city:* `char(20)`)
Supply(*sid:* `integer`, *pid:* `integer`)
Parts(*pid:* `integer`, *pname:* `char(20)`, *price:* `real`)

And assuming

- suppliers are 11500 tuples stored on 500 pages
- supply has 192000 tuples stored on 1500 pages
- parts table has 22400 tuples stored on 700 pages
- 35% of parts have a price less than 1000 (SF(prices<1000)= 0.35)
- 7% of suppliers from the city 'ELO' (SF(city='ELO')= 0.07)
- Assume the size of column types are: integer 4 bytes, char 1 byte, real 8 bytes
- The page size is 1024 bytes

Query1: SELECT S.sid, city
FROM Supply Y, Suppliers S
WHERE Y.sid=S.sid AND S.city='ELO'

Query1: SELECT pname, price
FROM Supply Y, Parts P
WHERE Y.pid=P.pid AND P.prices<1000

Question) For the given queries:

- List all possible execution plans.
- For each plan, estimate the size (in pages) of the intermediate results at every required level of the plan.
- Estimate the total I/O cost of each plan using one join algorithm of your choice.

Page size = 1024 bytes

- Sizes:
  `sid` = 4 B, `city` = 20 B → supplier tuple = 24 B
  `pname` = 20 B, `price` = 8 B → part tuple = 28 B

- Selection factors:
  - `city = 'ELO'` → SF = 0.07
  - `price < 1000` → SF = 0.35

Query 1 has 01 Plan Selection first, then join, selection in left side
Πsid, city(Πsid, city (σ(city='ELO')(Suppliers)))⋈(Πsid(Supply))

Estimation of the intermediate result sizes

Πsid, city(Πsid, city (σ(city='ELO')(Suppliers)))

- Suppliers filtered: Tuples=11500∗0.07=805 tuples
- Size per tuple = `sid` + `city` = 4 + 20 = 24 B
- Tuples per page = floor(1024 / 24) = 42 tuples per page
- Pages for filtered suppliers = ceil(805 / 42) ≈ 20 pages

Supply

- Supply = 192,000 tuples
- Size per tuple = `sid` = 4 B
- Tuples per page = floor(1024 / 4) = 256 tuples per page
- Pages for filtered suppliers = ceil(192000 / 256) ≈ 750 pages

Join cost using Block Nested Loop Join

500+20/(B-2) *750

Join cost using Block Nested Loop Join

SortΠsid, city(Πsid, city (σ(city='ELO')(Suppliers))) = 500 + 20 + $\log_{B-1}$ 20/B

Sort(Πsid(Supply)) = 1500+750+$\log_{B-1}$ 750/B

SortΠsid, city(Πsid, city (σ(city='ELO')(Suppliers))) + Sort(Πsid(Supply)) + [Πsid, city(Πsid, city (σ(city='ELO')(Suppliers)))] + [(Πsid(Supply))]=3540+ $\log_{B-1}$ 20/B + $\log_{B-1}$ 750/B

$$500 + 20 + \log_{B-1}\frac{20}{B} + 1500 + 750 + \log_{B-1}\frac{750}{B} + 20 + 750$$

$$\text{Total} = 3540 + \log_{B-1}\frac{20}{B} + \log_{B-1}\frac{750}{B}$$

$$\boxed{3540 + \log_{B-1}\frac{15000}{B^2}}$$

Query 1 has 02 Plan Selection first, then join, selection in left side

Πsid, city(Πpname, price (σ(prices<1000')(Parts)))⋈(Πpid(Supply))

Estimation of the intermediate result sizes

Πsid, city(Πpname, price (σ(prices<1000')(Parts))

- Suppliers filtered: Tuples=22400∗0.35=7840 tuples
- Size per tuple = pname+ price= 20 + 8 = 28 B
- Tuples per page = floor(1024 / 28) = 36 tuples per page
- Pages for filtered suppliers = ceil(7840 / 36) ≈ 217 pages

Supply

- Supply = 192,000 tuples
- Size per tuple = `pid` = 4B
- Tuples per page = floor(1024/4) = 256 tuples per page
- Pages for filtered suppliers = ceil(192000 / 256) ≈ 750 pages

Join cost using Block Nested Loop Join

700+217/(B-2)*750

Join cost using Block Nested Loop Join

Πsid, city(Πpname, price (σ(prices<1000')(Parts))) = 700+ 217 + $\log_{B-1}$ 217/B

Sort(Πpid(Supply)) = 1500+750+$\log_{B-1}$ 750/B

Πsid, city(Πpname, price (σ(prices<1000')(Parts))) + Sort(Πpid(Supply)) + [Πsid, city(Πpname, price (σ(prices<1000')(Parts)))] + [(Πpid(Supply))]=

$$700 + 217 + \log_{B-1}\frac{217}{B} + 1500 + 750 + \log_{B-1}\frac{750}{B} + 217 + 750$$

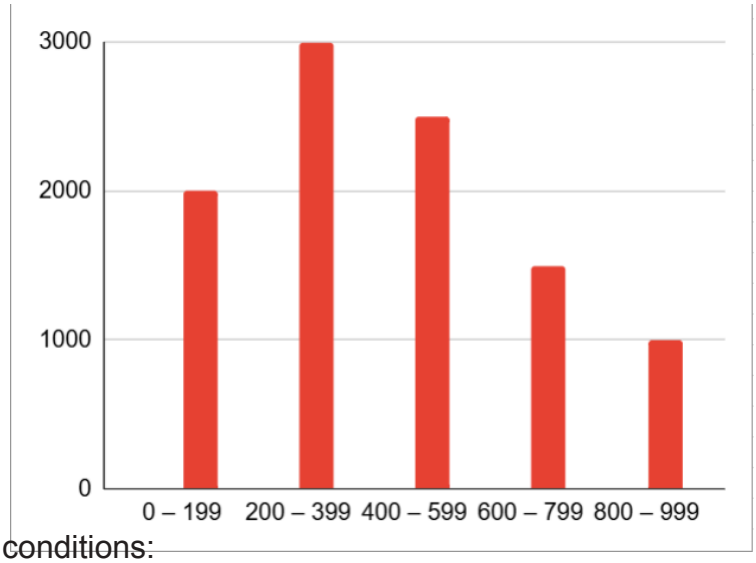
$$\boxed{4134 + \log_{B-1}\frac{162750}{B^2}}$$

NOTE: All other correct solutions are considred

Exercise 05 (3 marks, BONUS)

The Orders table has 10,000 rows, and we want to estimate the selectivity of a query condition using a histogram. Attribute order_amount has the following equally-width histogram with 5 buckets:

Bucket	Range	Frequency
1	0 – 199	2000
2	200 – 399	3000
3	400 – 599	2500
4	600 – 799	1500
5	800 – 999	1000



Estimate the selectivity factor for the following conditions:

1. order_amount = 350
2. order_amount < 500
3. order_amount BETWEEN 300 AND 700

1) order_amount = 350

350 in bucket 2 200 – 399 3000

bucket width=200

$$\text{Frequency per value} = \frac{3000}{200} = 15$$

$$\text{Selectivity factor} = \frac{15}{10000} = 0.0015$$

2) order_amount < 500

Bucket 1 (0–199) → fully included: 2000

Bucket 2 (200–399) → fully included: 3000

Bucket 3 (400–599) → partially included: 400–499 (100 values out of 200) → frequency proportion = 2500 × (100/200) = 1250

Total rows=2000+3000+1250=6250

Selectivity factor=6250/10000=0.625

3) order_amount BETWEEN 300 AND 700

Bucket 2 (200–399) → overlap 300–399 (100 values / 200) → frequency = 3000 × (100/200) = 1500

Bucket 3 (400–599) → fully included → frequency = 2500

Bucket 4 (600–799) → overlap 600–700 (101 values / 200) → frequency ≈ 1500 × (101/200) ≈ 757.5 ≈ 758

Total rows=1500+2500+758=4758

Selectivity factor=100004758≈0.476