Advanced Database Systems Winter Semester

Week 2- Part One (RAID Systems, Failvote, Message Passing)

Ahmad

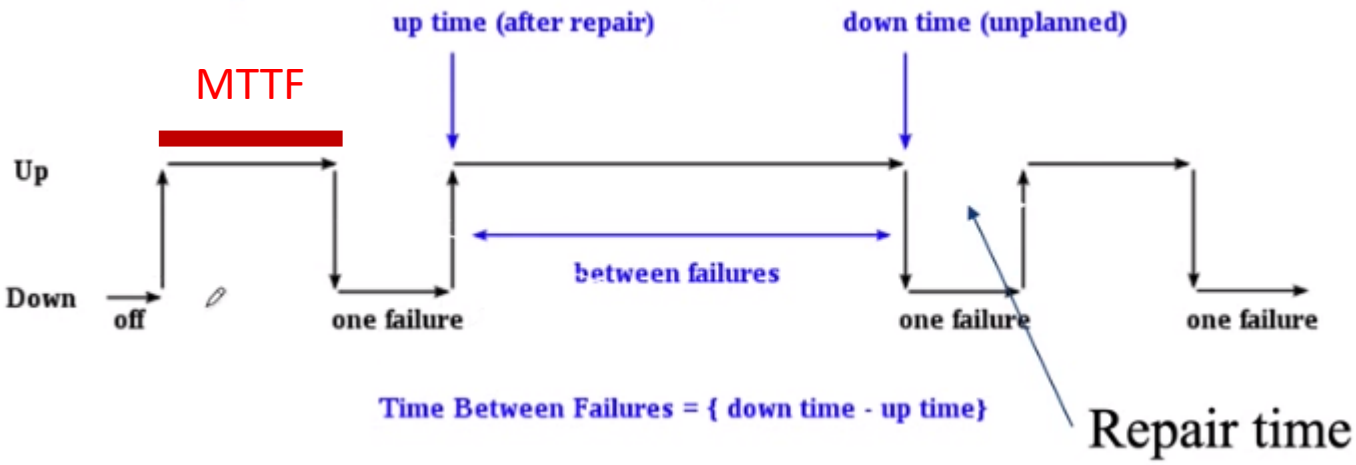
Question 1

- Which of the following RAID configurations that we saw in class has the lowest disk space utilization? Your answer needs to have explanations with calculations for each case.
- (a) RAID 0 with 2 disks
- (b) RAID 1 with 2 disks
- (c) RAID 3 with 3 disks
- Where does this lack of utilization of space go, i.e., where we can use such a configuration as it has some benefits gained due to the loss of space utilization?

A review

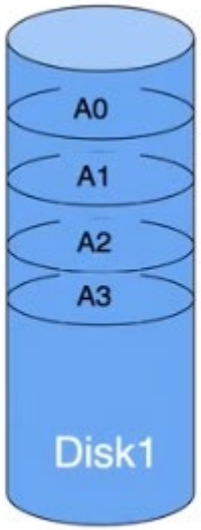
a) Mean time to failure:

- Definition
  - The time elapsed before a failure occurs.
- Show it on the Graph?
- If p is the probability of failure, how MTTF it calculated?
  - $1/p$



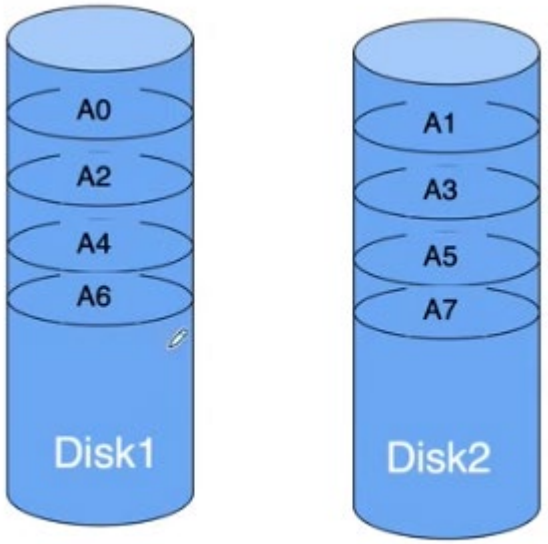
(Review) Baseline: One Disk

- MTTF: $1/p$
  - p: probability of failure of disk
  - $MTTF(disk) = \frac{1}{p}$



(Review) Raid 0: Separated Storage

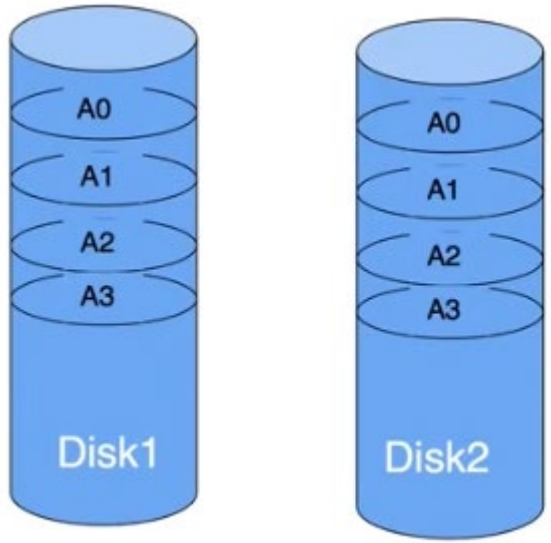
- Advantages:
  - Double throughput Compared to One Disk
- Disadvantage:
  - Shorter MTTF
  - Fails if one of the disks fail



$$P(failure) = P(A) + P(B)$$
$$P(failure) = 2p$$
$$MTTF(Raid_0) = \frac{1}{2p} = \frac{1}{2} \times MTTF(disk)$$

Raid 1: Mirrored Storage

- Data is mirrored across the two storages
- Disadvantages:
  - Double Storage
- Advantages: Better Fault Tolerance
  - How much better?



Raid 1: Mirrored Storage

- Data is mirrored across the two storages
- Disadvantages:
  - Double Storage
- Advantages: Better Fault Tolerance
  - How much better?
    - Fails if both of them fail.

$$P(failure) = P(A \,\&\, B)$$
$$P(failure) = P(A) \times P(B)$$
$$P(failure) = p^2$$
$$MTTF(RAID_1) = \left(\frac{1}{p}\right)^2 = MTTF(disk)^2$$

What about Raid 1 with 3 discs?

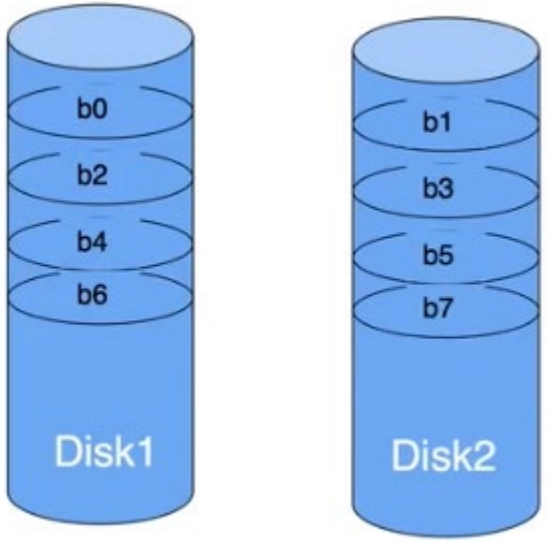
- Data is mirrored across the two storages
- Disadvantages:
  - Double Storage
- Advantages: Better Fault Tolerance
  - How much better?
    - Fails if both of them fail.

$$P(failure) = P(A \,\&\, B)$$
$$P(failure) = P(A) \times P(B)$$
$$P(failure) = p^2$$
$$MTTF(RAID_1) = \left(\frac{1}{p}\right)^2 = MTTF(disk)^2$$

What about Raid 1 with 3 discs?

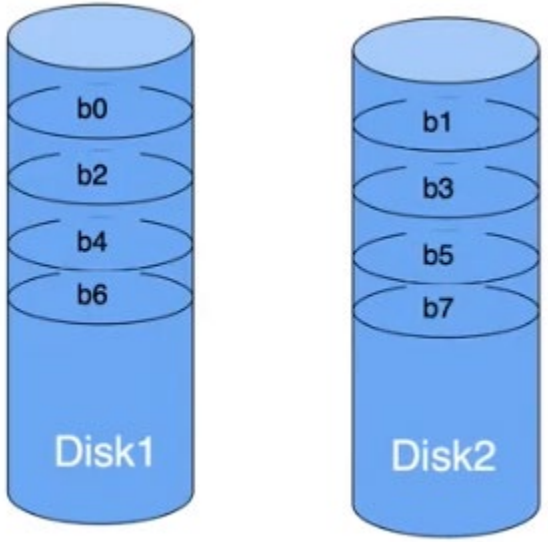
Raid 2: Bit-level separation

- Similar to Raid 0, but stores data at the bit level.
- Advantages:
  - Higher transfer rate
- Disadvantages
  - Rarely used
  - what about MTTF?



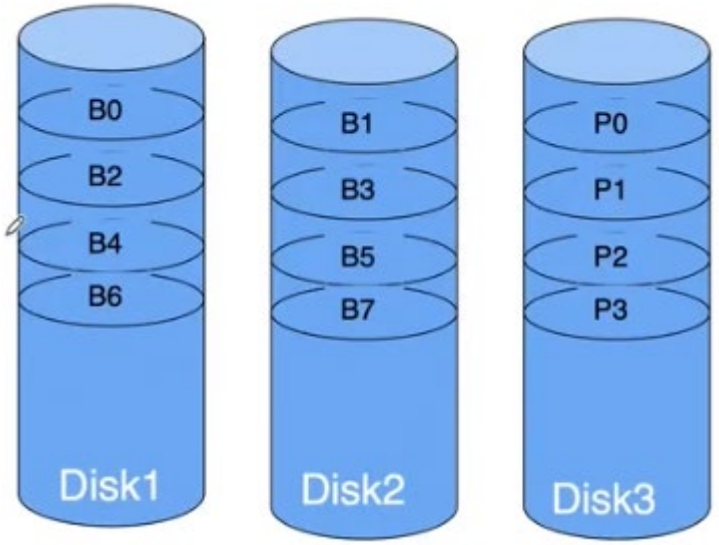
Raid 2: Bit-level separation

- Similar to Raid 0, but stores data at the bit level.
- Advantages:
  - Higher transfer rate
- Disadvantages
  - Rarely used
  - what about MTTF?
    - Is same as the one for Raid 0.



Raid 3:Byte Parity Raid

- Data stored separately, and a separate parity bit is also stored for each bit
- Advantages:
  - Higher transfer rate than Raid 0.
- Disadvantages:
  - Rarely used.
- What is the MTTF?
  - Breakout rooms.



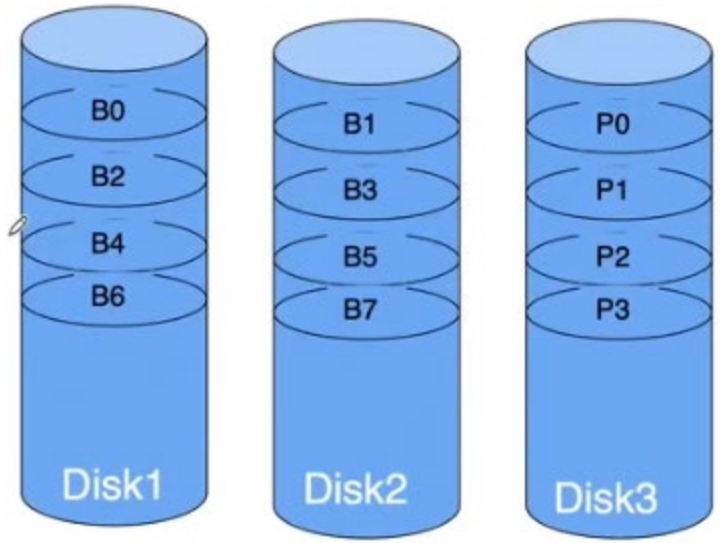
Raid 3:Byte Parity Raid (MTTF)

- Fails, if any two of three disks fail.

$$P(failure) = \binom{3}{2} p \times p = 3p^2$$

$$MTTF(Raid_3^3) = \frac{1}{3} \times \left(\frac{1}{p}\right)^2 = \frac{1}{3} MTTF(disk)^2$$

- $MTTF(Raid_3^3)?\, MTTF(Raid_1)$
  - Which one is better?



Raid 3:Byte Parity Raid (MTTF)

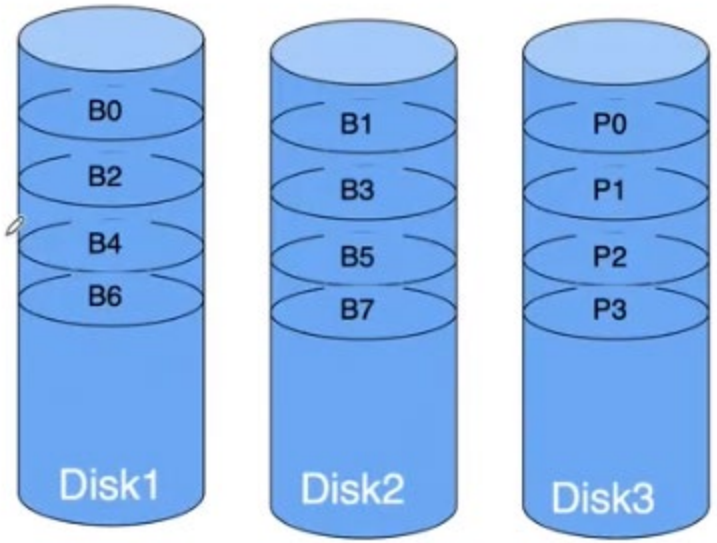
- Fails, if any two of three disks fail.

$$P(failure) = \binom{3}{2} p \times p = 3p^2$$

$${}^{n}C_{r} = \frac{n!}{(n-r)!\, r!}$$

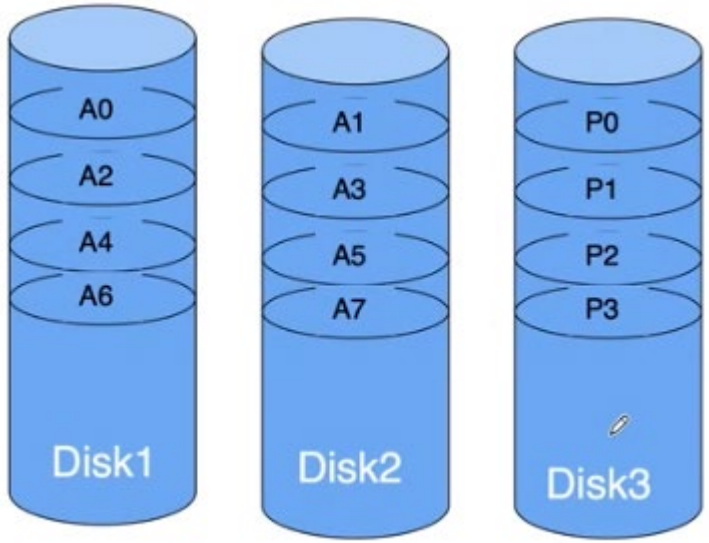
$$MTTF(Raid_3^3) = \frac{1}{3} \times \left(\frac{1}{p}\right)^2 = \frac{1}{3} MTTF(disk)^2$$

- $MTTF(Raid_3^3)?\, MTTF(Raid_1)$
  - Which one is better?
  - $MTTF(Raid_3^3) = \frac{MTTF(Raid_1)}{3}$
  - $Raid_1$ is better.



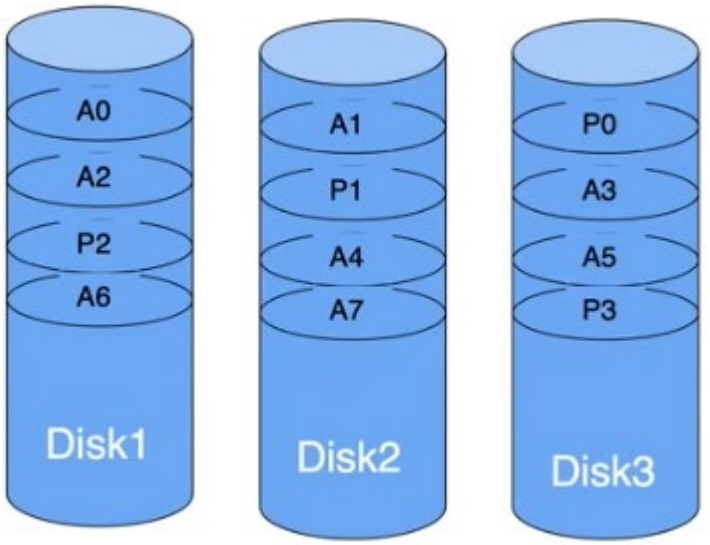
Raid 4

- Similar to Raid 3, but stored at Block Level
- Advantages:
  - Higher throughput.
- Disadvantages:
  - Very slow writes
    - Disk 3 has more writes, as parity needs to be updated.
    - How to handle this?
- What is the MTTF?



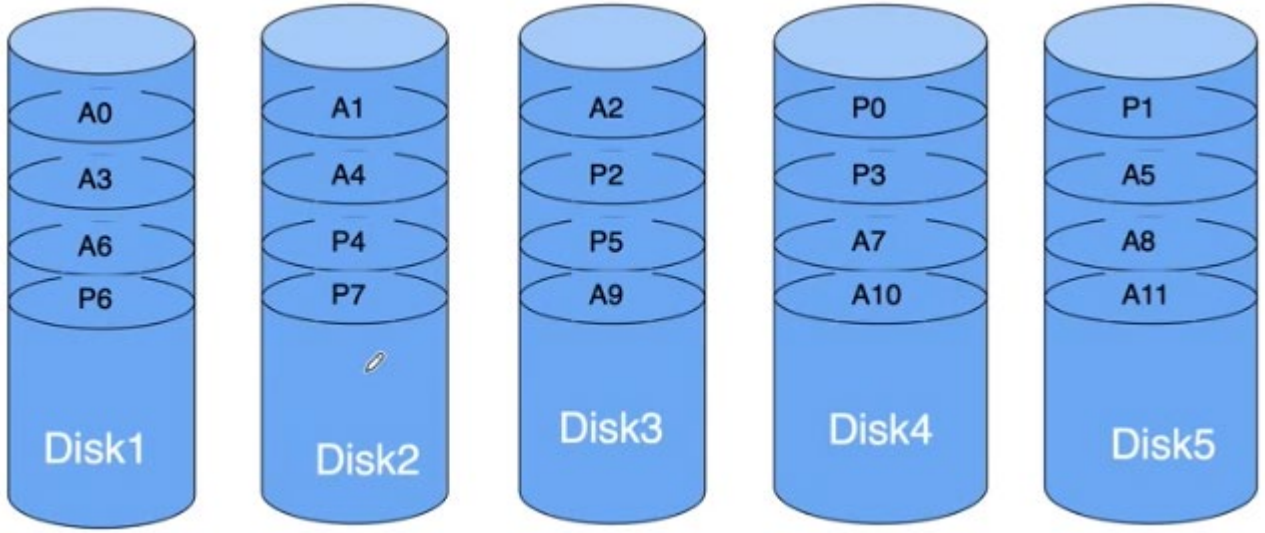
Raid 5

- Similar to Raid 4, but Parity is distributed among all disks.
- Advantages:
  - Higher Throughput
  - Better write speed compared to Raid 4
- What is the MTTF for Raid 5?



Raid 6:

- Extension of Raid 5 with two more disks
- Parities are stored in a way to insure recovery in the case of two failures.
- Advantages:
  - Higher Throughput
  - Longer MTTF
- What is MTTF?



Raid 6: MTTF (Breakout rooms)

$$P(Failure) = P(A\&B\&C) + P(A\&B\&D) + \ \ldots$$

$$P(Failure) = \binom{5}{3} p^3 = 10p^3$$

$$MTTF = \frac{1}{10} \times \left(\frac{1}{p}\right)^3 = \frac{1}{10} MTTF(disk)^3$$

Question 1

- Which of the following RAID configurations that we saw in class has the lowest disk space utilization? Your answer needs to have explanations with calculations for each case.
- (a) RAID 0 with 2 disks
- (b) RAID 1 with 2 disks
- (c) RAID 3 with 3 disks
- Where does this lack of utilization of space go, i.e., where we can use such a configuration as it has some benefits gained due to the loss of space utilization?

Question 1- Solution

- (a) RAID 0 with 2 disks
  - In case 1, the space utilization is 100% because the two disks store contiguous blocks of a file in RAID 0.
- (b) RAID 1 with 2 disks
  - In case 2, the space utilization is 50% because RAID 1 uses mirroring. MTTF increases so for cases where a disk can fail easily this is good. The system operates even when a disk fails.
- (c) RAID 3 with 3 disks
  - In case 3, the space utilization is (3-1)/3=66.7 because RAID 3 uses one disk for storing parity data.
- Therefore case 2 has the lowest disk utilization.

Question 2

- We start with question 2. What is the mean time to failure value for different RAID systems, please review/discuss?

  a. RAID 0 with 2 disks

  b. RAID 2 with 2 disks

  c. RAID 1 with 2 disks

  d. RAID 1 with 3 disks

  e. RAID 3 with 3 disks

  f. RAID 4 with 3 disks

  g. RAID 5 with 3 disks

  h. RAID 6 with 5 disks

Voting when failure happens

- When we receive inconsistent readings from different disks, what action should be taken?
  - A subset of disks read 2, while the rest read 5.
- Majority voting is the solution to inconsistent reads or any other actions where there is not a consensus.
- Two types of majority voting:
  - Failvote: Majority of all systems need to commit.
  - Failfast: Only a majority of working systems need to commit.
- In both cases, define majority intuitively, as half plus one or: $\left\lfloor \frac{n}{2} \right\rfloor + 1$.

Question 3

- In a Failvote system, in which of the following cases can we accept an action?

Total number of devices	Number of agreeing devices	Accept?
10	6	
10	5	
10	4	
5	3	
5	2	

Question 5- Solution

- In a Failvote system, in which of the following cases can we accept an action?

Total number of devices	Number of agreeing devices	Accept?
10	6	y
10	5	n
10	4	n
5	3	y
5	2	N

Question 4

- In a Failfast system, in which of the following cases can we accept an action?

Total number of devices	Number of working devices	Number of agreeing devices	Accept?
10	6	4	
10	6	3	
10	5	3	
5	5	3	
5	4	2	
5	2	2	
5	1	-	

Question 4- Solution

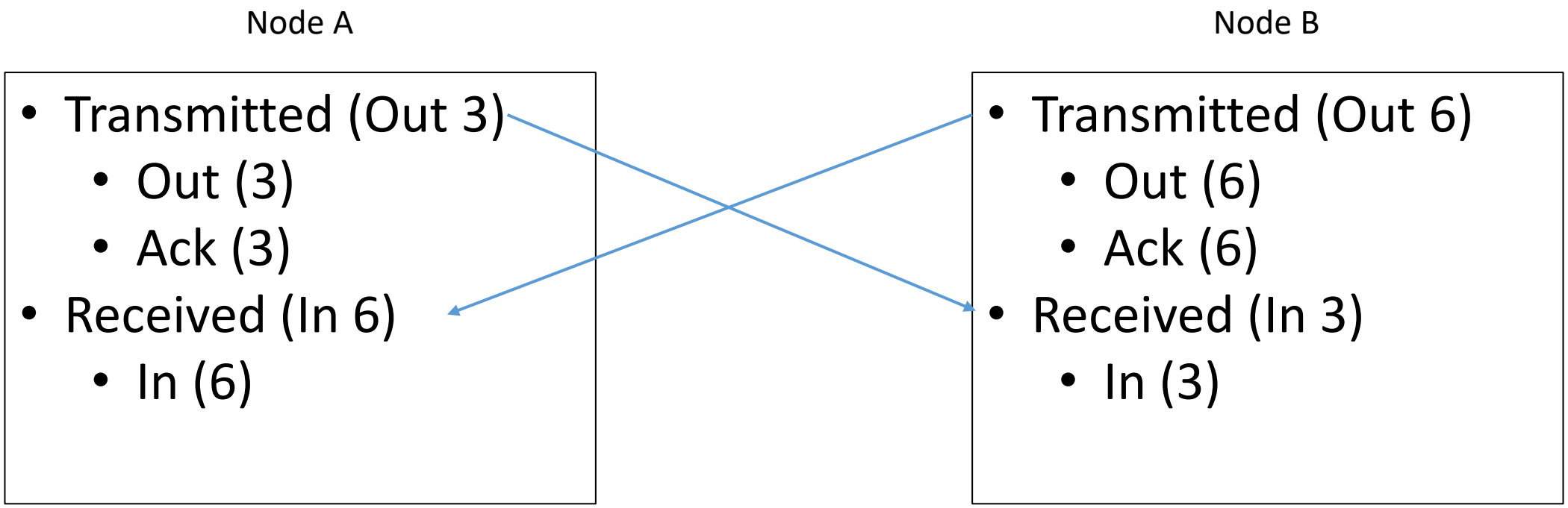
- In a Failfast system, in which of the following cases can we accept an action?

Total number of devices	Number of working devices	Number of agreeing devices	Accept?
10	6	4	y
10	6	3	n
10	5	3	y
5	5	3	y
5	4	2	n
5	2	2	y
5	1	-	n

Question 5

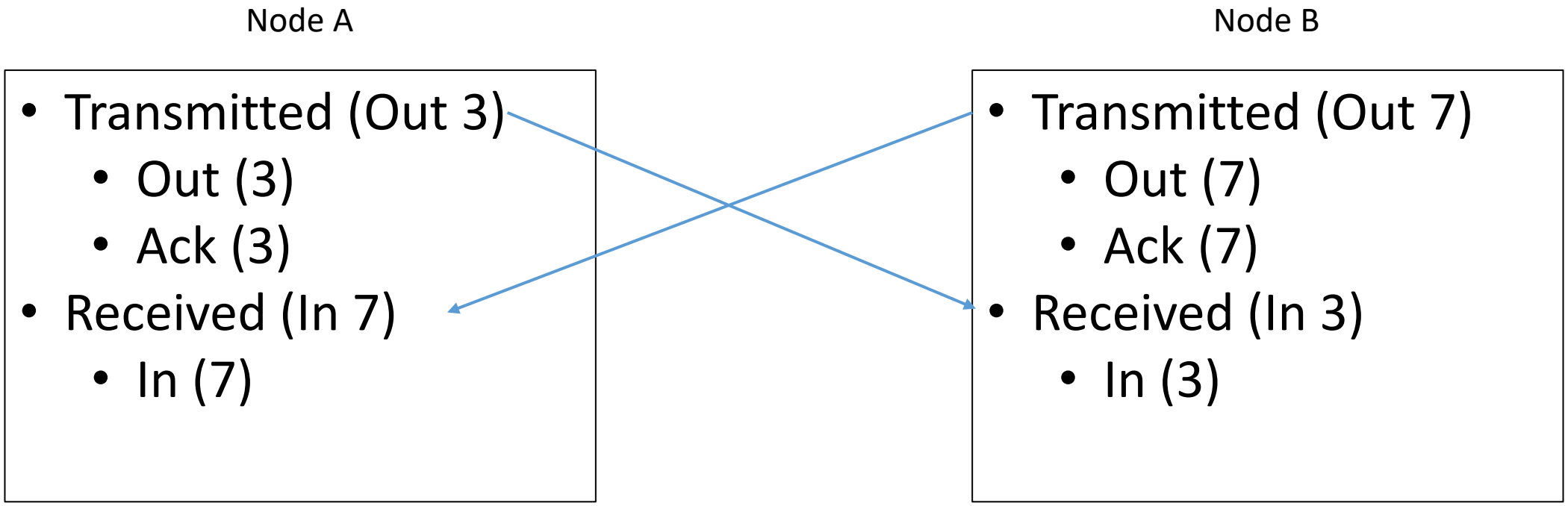
- There are two nodes in a network that use stable storage and acknowledgment message passing for reliable communication. The stable storage of Node A contains the following record - Received message (In6); Transmitted message(Out3); Out:3 Ack:3 In:6. The stable storage of Node B contains the following record - Received message (In3); Transmitted message(Out6); Out:6 Ack:6 In:3.
- Now Node B sends a new message 7 to Node A. What will be in the stable storage of A and B if the message is received correctly, including a correctly received acknowledgement?

Solution to Question 5



Solution to Question 5

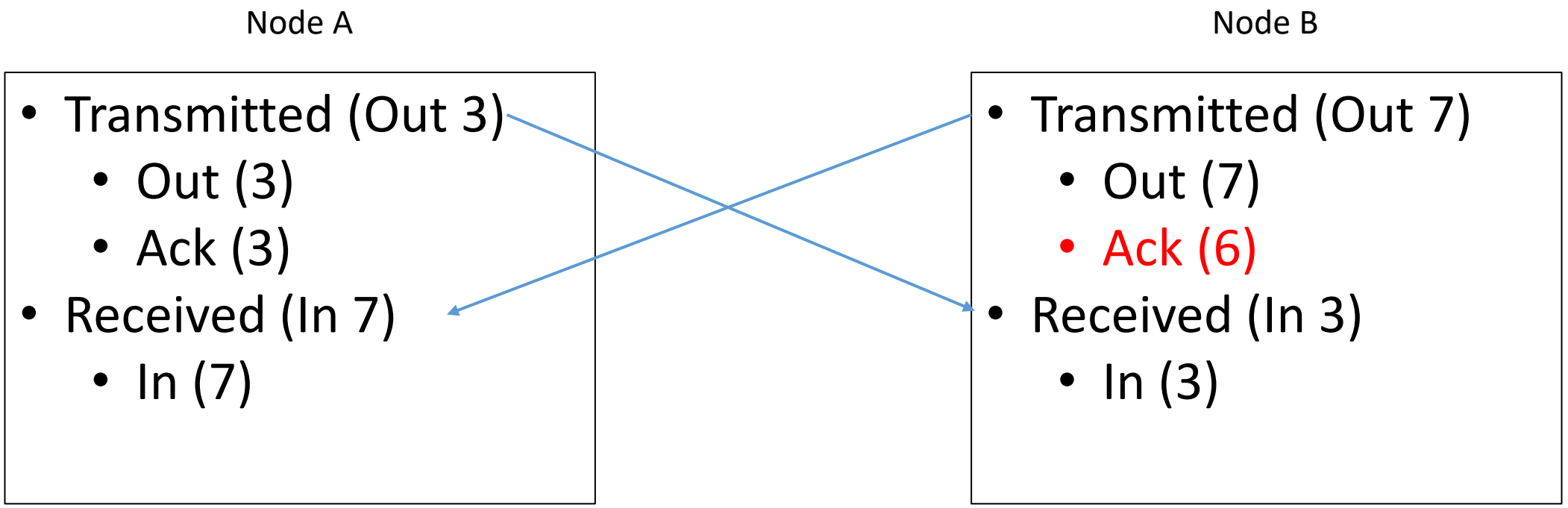
Now Node B sends a new message 7 to Node A. What will be in the stable storage of A and B if the message is received correctly, including a correctly received acknowledgement?



Question 6

- Assume The initial State as of question 3
- Now Node B sends a new message 7 to Node A. What will change in the stable storage of A and B if the message is received correctly, but the acknowledgement is not received?

Solution to Question 6



Advanced Database Systems Winter Semester

Week 2- Part Two (Query Optimization, Nested Loop Join)

Ahmad

Strategies for Query Optimizations

- Searching/Enumerating all the plans and choose the best one.
  - Used for quires that require accurate results.
  - Well suited for handling complex quires.
- Using a heuristic approach
  - Is like using a solution randomly.
  - Is a good option when accuracy is not a priority.
  - Can handle simple and straight-forward cases.

Question 7

- Which of the query optimization approaches is suitable for the following queries?
- Scenario A: (Given a table with 10 million tuples, run a query)

  SELECT customer

  FROM Table

  WHERE spend BETWEEN 100 AND 200

  AND birth_year> 2000

Question 7-A, Solution

- Which of the query optimization approaches is suitable for the following queries?
- Scenario A: (Given a table with 10 million tuples, run a query)

  SELECT customer

  FROM Table

  WHERE spend BETWEEN 100 AND 200

  AND birth_year> 2000

- **Solution:** For Scenario A, the heuristic approach would be suitable due to the simplicity of the query.

Question 7- B

- Scenario B: (Given 5 tables with 1000 tuples in each table, run a query)

```
SELECT T1.name, T2.salary, T3.qualification, T4.phone, T5.leader
FROM Table1 T1
    INNER JOIN Table2 T2 ON T2.id = T1.id
    INNER JOIN Table3 T3 ON T3.id = T1.id
    INNER JOIN Table4 T4 ON T4.id = T1.id
    INNER JOIN Table5 T5 ON T5.department = T1.department
WHERE T1.age > 50;
```

Question 7- B, Solution

- Scenario B: (Given 5 tables with 1000 tuples in each table, run a query)

```
SELECT T1.name, T2.salary, T3.qualification, T4.phone, T5.leader
FROM Table1 T1
    INNER JOIN Table2 T2 ON T2.id = T1.id
    INNER JOIN Table3 T3 ON T3.id = T1.id
    INNER JOIN Table4 T4 ON T4.id = T1.id
    INNER JOIN Table5 T5 ON T5.department = T1.department
WHERE T1.age > 50;
```

Solution: For Scenario B, the enumerating approach would be more suitable due to the complexity of the query.

About Nested-Loop Join

- To compute a theta join

  for each tuple t_r **in** r **do begin**

  for each tuple t_s **in** s **do begin**

  test pair (t_r, t_s) to see if they satisfy the join condition theta (θ)

  if they do, add t_r t_s to the result.

  end

  end

- The number of block transfers is $n_r * b_s + b_r$.
- The number of seeks is $n_r + b_r$

Example with Nested Loop Join

- n_r=3000000, $b_r = 1000$ and $n_s = 400, \mathrm{b_s} = 10$
  - $block\ transfers = \ n_r * b_s + b_r = 3000000 \times 10 + 1000 = 30{,}001{,}000$
  - $seek\ times = n_r + b_r = \ 3000000 + 1000 = 3001000$

  - average seek time is $12\ ms$, means $3001000 \times 12\ ms \sim 1\ \mathrm{hr}$ for a simple join.

- How to improve this?
  - Match block by block instead to minimize the number of seek times.

Improved Nested-Join; Block Nested-Join

```
for each block B_r of r do begin
    for each block B_s of s do begin
        for each tuple t_r in B_r do begin
            for each tuple t_s in B_s do begin
                Check if (t_r,t_s) satisfy the join condition
                if they do, add t_r • t_s to the result.
            end
        end
    end
end
```

- Number of block transfers: $b_r \times b_s + b_r$
- Number of seeks: $2 \times b_r$

Example with Block Nested-Loop Join

- Number of seeks = $2 \times b_r = 2000$
- The new seek time is $2000 \times 12\ ms \sim 24\ seconds$

- In orders three orders of magnitude less than the Nested-Loop join.

Question 8

- Review the examples on nested-loop join and block nested-loop join given in the lecture. Discuss and calculate why the later one can be more efficient.

Question 8- Solution

- Review the examples on nested-loop join and block nested-loop join given in the lecture. Discuss and calculate why the later one can be more efficient.

  - After reiterating the examples/calculations, we see the later one is most efficient because each block in the inner relation is read once for each block in the outer relation (instead of once for each tuple in the outer relation)

Question 9

- A particular query on table A used to run quite efficiently in a DBMS. After inserting many records and deleting many other records from table A, that same query is now taking more time to run, even when the total number of records has not changed. What can be the reason for that? What can you do as the user/database administrator of that DBMS to improve the performance of this query?

Question 9- Answer

- Solution:
- After insertions and deletions, the statistics of a table may not be updated instantly. As the statistics are used to estimate the cost of a query, wrong statistics are causing wrong cost estimations, and hence the query optimiser is now choosing a query plan that is no longer optimal for that query.
- Enforcing statistical recompilation option can be used to update the statistics of the table.

Advanced Database Systems Winter Semester

Week 2- Part Three (Data Structures for Big Data)

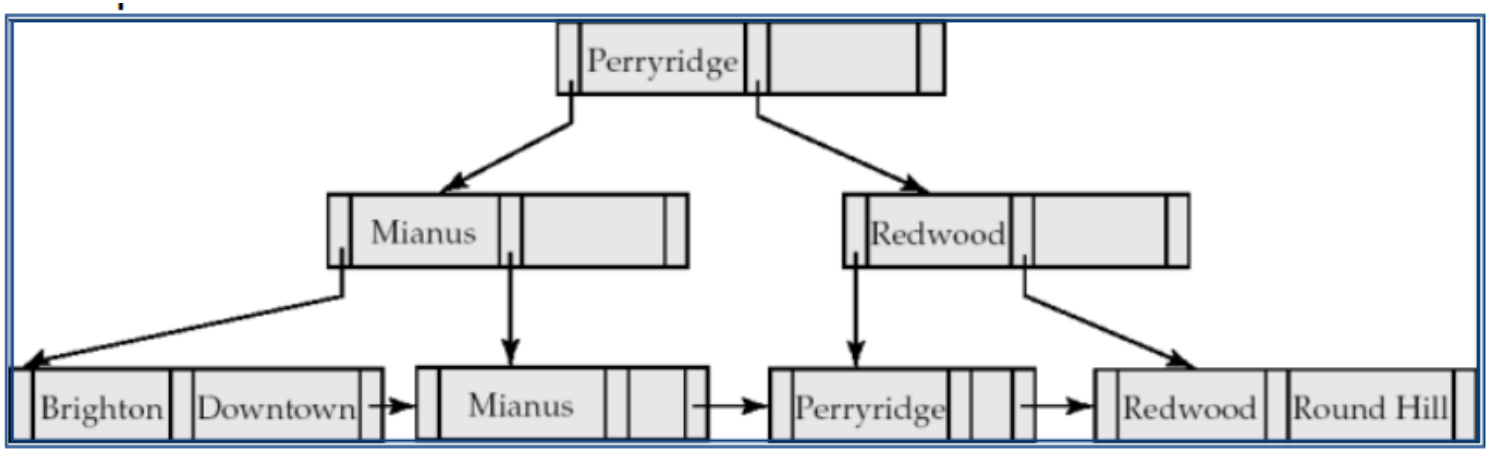
Ahmad

Are Heuristic and Enumerating the only options?

- Systems may use heuristics to reduce the number of choices that must be made in a cost-based fashion
- However, DBMS admin generally creates indices to allow almost direct access to individual items
- Two basic kinds of indices:
  - Ordered indices: search keys are stored in some order
  - Hash indices: search keys are distributed hopefully uniformly across “buckets” using a “function”
- The impacts:
  - Faster disk access; but insertion and deletion is also important.

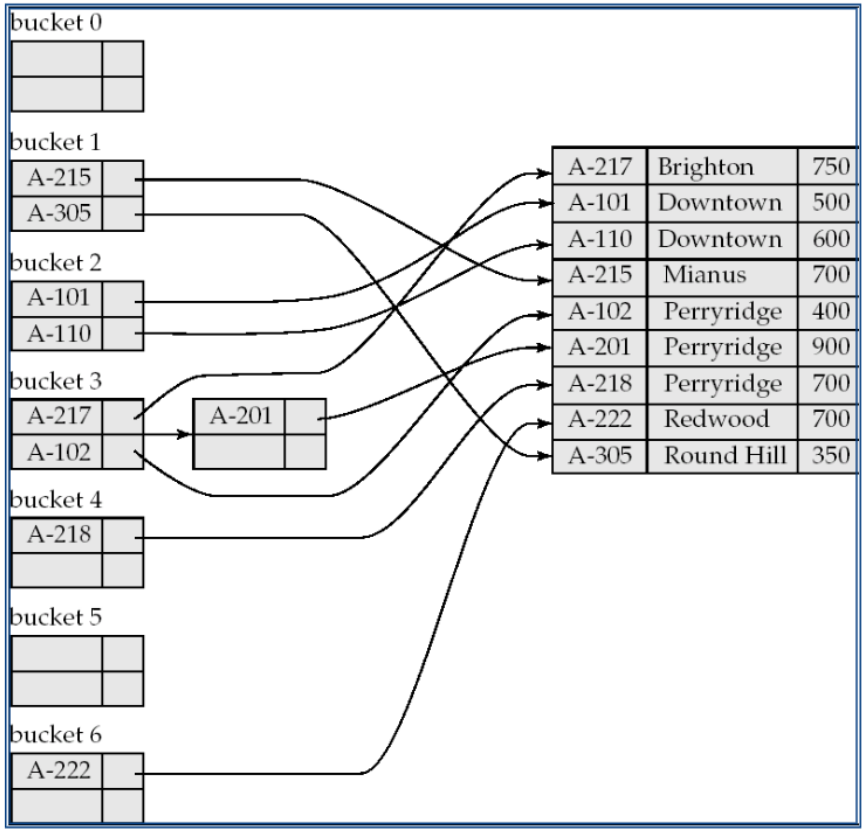
Most Popular Index in DMBS: B+Tree

- Features:
  - Keeps the data in order
  - Enables Binary Search
  - Maintained by periodic reorganization
  - If there are K search-key values in the file, the height of the tree is no more than $\log_{n/2}(K)$ and it would be balanced.
  - Perfect for Range queries.



Other forms of indices: Hash Indices

- Unordered, but record pointers are used with search keys.
- Given a key the aim is to find the related record on file in one shot which is important.
- An good hash function is uniform (same per buc)
- Ideal hash function is random



Bitmap Indices

- Records in a relation are assumed to be numbered sequentially from, say, 0
- Applicable on attributes that take on a relatively small number of distinct values
  - E.g. gender, country, state, ...
  - E.g. income-level (income broken up into a small number of levels such
  - as 0-9999, 10000-19999, 20000-50000, 50000- infinity)
- A bitmap is simply an array of bits.

R-Trees

- R-trees are a N-dimensional extension of B+-trees, useful for indexing sets of rectangles and other polygons.
- Idea: generalize the notion of a one-dimensional interval associated with each B+ -tree node to an N-dimensional interval, that is, an N-dimensional rectangle.

Question 10 (Breakout Rooms)

- What indices are suitable if a table is frequently used for finding records based on three criteria?
  - a list of users' name,
  - a range of users' birthday
  - a spatial region covering users' residence

Question 10 - Solution

- What indices are suitable if a table is frequently used for finding records based on three criteria?
  - a list of users' name,
  - a range of users' birthday
  - a spatial region covering users' residence

Solution:

- Users' name: Hash index.
- Users' birthday: B+ tree index.
- Users' residence: R-tree index or another spatial index such as a quadtree index.

Question 11- Breakout Rooms

- Review the points on indexing with B+ trees. Assume a database table has 10,000,000 records and the index is built with a B+ tree. The maximum number of children of a node, is denoted as n.
  - How many steps are needed to find a record if n = 4?
  - How many steps are needed to find a record if n = 100?

Question 11- Solution

- Review the points on indexing with B+ trees. Assume a database table has 10,000,000 records and the index is built with a B+ tree. The maximum number of children of a node, is denoted as n.
  - How many steps are needed to find a record if n = 4?
  - How many steps are needed to find a record if n = 100?

Solution:

- When $n = 4$, the maximum height of the tree is

$$\lceil \log_{\lceil n/2 \rceil}(K) \rceil = \lceil \log_2(10000000) \rceil = 24$$

  Therefore, 24 steps are needed.
- When $n = 100$, the maximum height of the tree is

$$\lceil \log_{\lceil n/2 \rceil}(K) \rceil = \lceil \log_{50}(10000000) \rceil = 5$$

  Therefore, 5 steps are needed.

Search in R-Trees

- To find data items intersecting a given query point/region, do the following, starting from the root node:
  - If the node is a leaf node, output the data items whose keys intersect the given query point/region.
  - Else, for each child of the current node whose bounding box intersects the query point/region, recursively search the child.

Can be very inefficient in worst case since multiple paths may need to be searched due to overlaps, but works acceptably in practice.

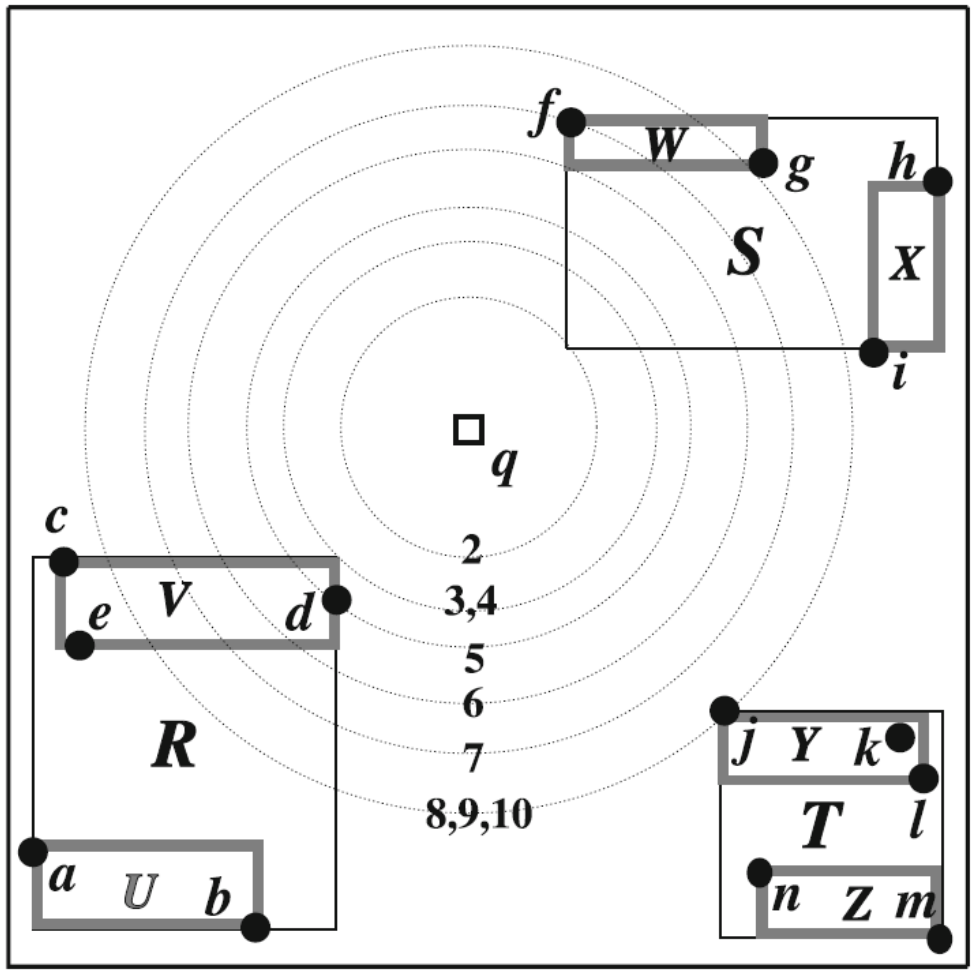
Nearest Neighbour in R-Trees

To find the nearest neighbour of a given query point/region, do the following, starting from the root node:

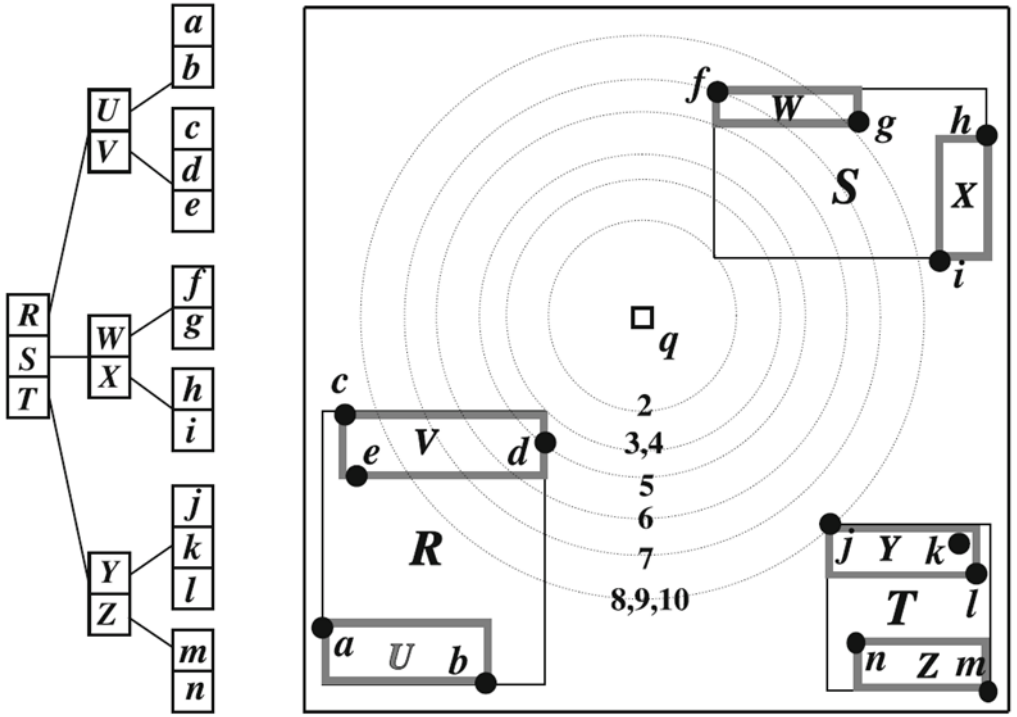
- Use a sorted priority queue of the R-tree nodes based on the minimum distance from the query
- Traverse the node that is in the top of the priority queue, and put its elements in the queue. Continue
- Stop when the top node is a data object (first NN has been found)

This algorithm is a best-first search algorithm

Example R-tree



Nearest Neighbor Query on R-tree

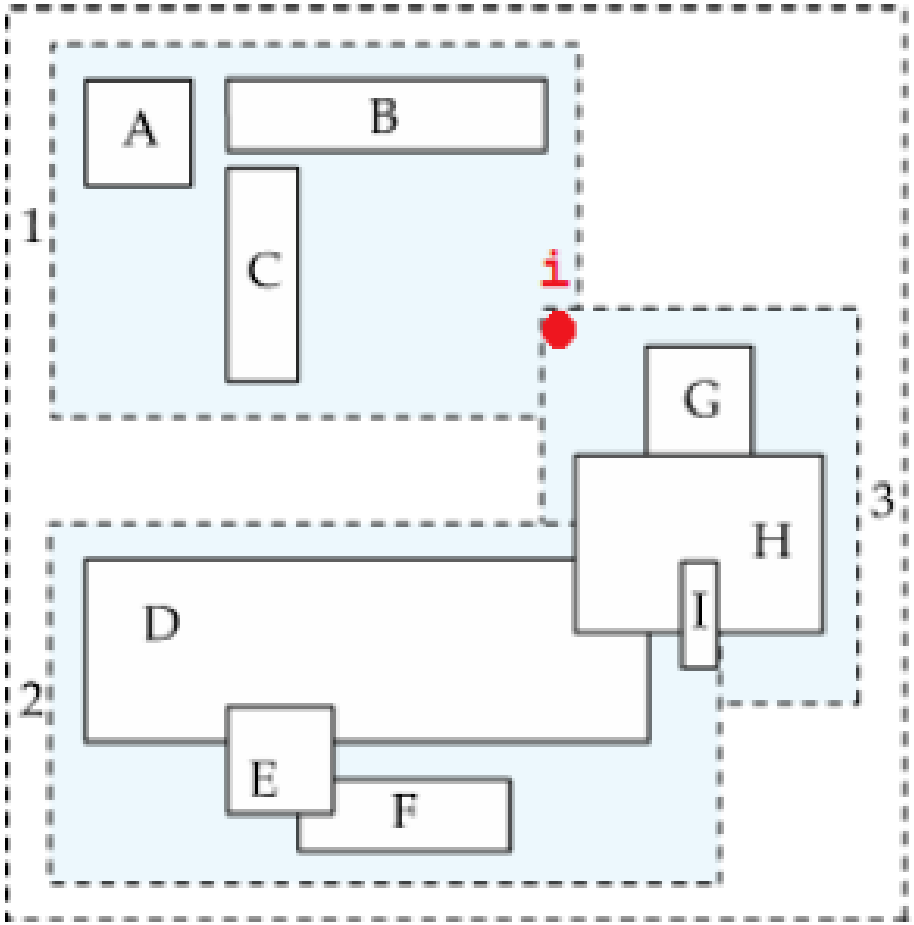


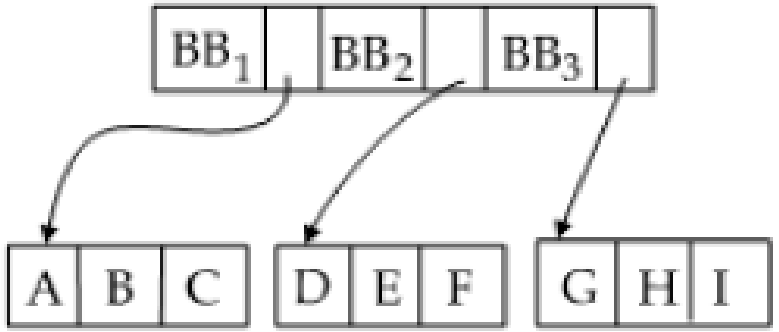
Step	Priority queue	Retrieved NN(s)
1	⟨*S*, *R*, *T*⟩	⟨⟩
2	⟨*R*, *W*, *T*, *X*⟩	⟨⟩
3	⟨*V*, *W*, *T*, *X*, *U*⟩	⟨⟩
4	⟨***d***, *W*, *T*, *X*, ***c***, ***e***, *U*⟩	⟨⟩

Step 5 finds d as the first NN (using Best First search)

Question 12- Search the closest to “i” in R-Tree?

Anything special?





Question 12- Solution

- Solution:
- In this traversal we first visit node BB1 as it overlaps with the query point, and find that object B is the closest to query point i.
- The issue here is that we cannot stop at this point in the traversal as i overlaps with BB3 as well.
- After investigating BB3, we figure out that G is the closest object overall.

Due to overlaps in R-tree branches two or more branches of an R-tree need to be investigated in many query types. In addition, as each internal node represents a bounding box, thus we are not sure about the position of objects in a bounding box which may necessitate that we investigate multiple bounding boxes to determine a nearest neighbour in this case.