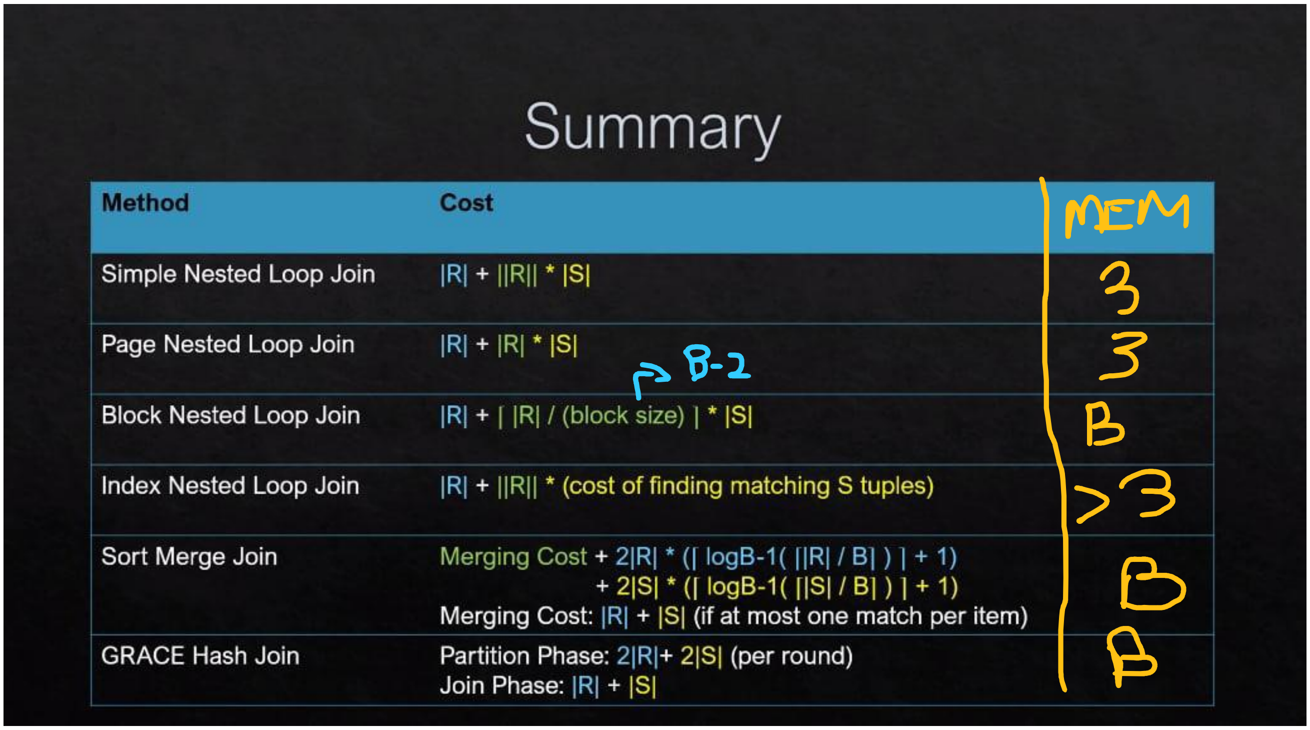
AP formula: Sn = (n/2)[2a + (n-1)(d)]
KB/MB/GB = 2^{10}/ 2^{20}/ 2^{30} B
Rotational Delay = $\frac{60}{rpm} \cdot 10^3\ ms$
Avg Rotational Delay = $\frac{Rotational\ Delay}{2}$
Time to Read 1 Sector with Gap = $\frac{Rotational\ Delay}{number\ of\ sectors}\ ms$
Time to Read 1 Sector **without** Gap = $\frac{Time\ to\ read\ 1\ sector\ with\ gap}{1-(percentage\ of\ gap\ on\ track)}\ ms$
Transfer Time = (# sectors w gap * Time read w gap) + ((#sectors - #sectors w gap) * Time read w/o gap)



Summary

Method	Cost	MEM
Simple Nested Loop Join	|R| + ||R|| * |S|	3
Page Nested Loop Join	|R| + |R| * |S|	3
Block Nested Loop Join	B-2 |R| + ⌈ |R| / (block size) ⌉ * |S|	B
Index Nested Loop Join	|R| + ||R|| * (cost of finding matching S tuples)	>3
Sort Merge Join	Merging Cost + 2|R| * (⌈ logB-1(⌈|R| / B⌉) ⌉ + 1) + 2|S| * (⌈ logB-1(⌈|S| / B⌉) ⌉ + 1) Merging Cost: |R| + |S| (if at most one match per item)	B
GRACE Hash Join	Partition Phase: 2|R|+ 2|S| (per round) Join Phase: |R| + |S|	B

*Note: GRACE Hash Join formula only applies if $B \geq \sqrt{number\ of\ pages}$, if not repeatedly partition

(to be exact: $\frac{M}{B-1} \leq B-2$)

Repeated partition of hash join

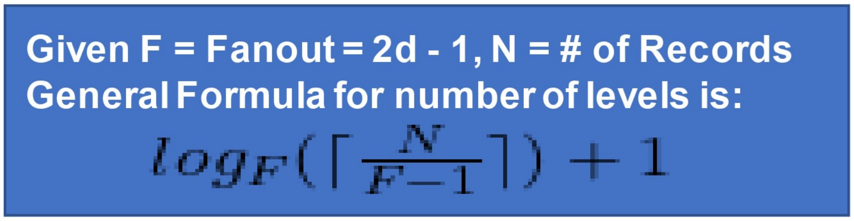
- Incur 2|R| + 2|S| for each additional round
- New number of pages = M/(B-1)$^{\text{no. of additional partition}}$

Hash-Index

- Format 1: 1.2 (1 if no overflows)
- Format 2: 1.2 (1 if no overflows) + Cost of finding S tuples (depends on no. of matching page/tuples for clustered/unclustered)

B-trees

- Finding Order of B+ tree, d: (2d+1)(ptr_size) + (2d)(key_size) <= page_size
  - 'd' will round down, m = 2d (m must be divisible by 2) ***implication: m must be an even number
  - Height of B+ Tree = ↱ $\log_m$ $total\ number\ of\ records$ ↰

- 

Given F = Fanout = 2d - 1, N = # of Records
General Formula for number of levels is:

$$log_F(\lceil \frac{N}{F-1} \rceil) + 1$$

- For a n-level B+-tree, the <u>minimum</u> number of data entries is $2d(d+1)^{n-2}$ [1-level tree: d]
- For a n-level B+-tree, the <u>maximum</u> number of data entries is $4d(d+1)^{n-2}$ [1-level tree: 2d]
- Total cost = Cost for format 1/2/3 + Cost for clustered/unclustered (for format 2/3) + Additional leaf node if data entries across multiple nodes?
  - Cost for format 1/2/3: height of the tree (includes first leaf node) [index is in memory]
  - Cost for clustered: Each data <u>page</u> is only fetched once (1 I/O per required page)
  - Cost for unclustered: Each data <u>record</u> is fetched (1/O per required data tuple/pointer)
- d <= keys <= 2d only applies for format 2, format 1 and 3 (except for leaf nodes)

Multi-way Merge Sort:

- Size of 1 run: Usually B
- No. of passes = $1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil$ (If generating runs included) *** alternatively divide manually since this formula somehow gives weird answers sometimes
- Cost = 2N*(No. of passes)
- To merge in one pass, #sorted runs <= number of available input buffers

Replacement Selection (freeze if new value <= old value)

- Average length of a run: 2B
- Min length of a run: B (Worst case: when file is sorted in descending order) -> same as quicksort
- Max length of a run: M (Best case: when file is sorted in ascending order)

Sort-Merge Join

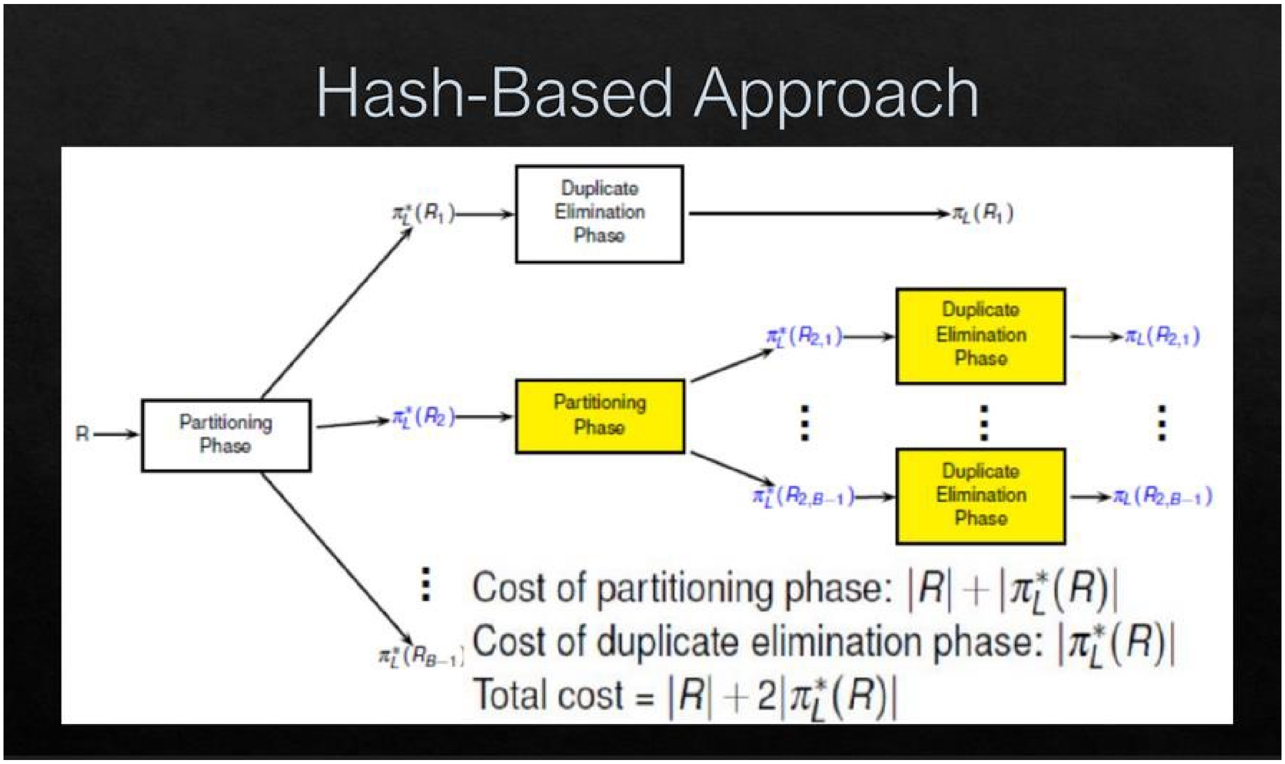
- Best Case: Largest value in R < Smallest value in S => Merging Cost: |R| + 1 ***when R and S near sorted/already sorted?
- Worst Case: Each tuple of R requires scanning entire S => Merging Cost: |R| + |R|*|S|

GRACE Hash Join

- Min no. of partitions during partition phase:
  - Each partition uses B-1 pages (1 reserved for input)
  - No. of partitions = M/B-1
- Max no. of partitions during partition phase:
  - Each partition uses only M/B-1 page
  - No. of partitions = B-1

Optimized Sort-based Approach

- An approach based on sorting:
  - Modify phase 1 of external sort to eliminate unwanted fields
    - Runs are produced, but tuples in runs are smaller than input tuples (Size ratio depends on # and size of fields that are dropped)
  - Modify merging passes to eliminate duplicates
    - Number of result tuples smaller than input (Difference depends on # of duplicates)
  - Cost:
    - In phase 1, read original relation (size M), write out same number of smaller tuples
    - In merging passes, fewer tuples written out in each pass



Hash-Based Approach

Cost of partitioning phase: $|R| + |\pi_L^*(R)|$

Cost of duplicate elimination phase: $|\pi_L^*(R)|$

Total cost = $|R| + 2|\pi_L^*(R)|$