ETL

-Extract (read relevant data)

-Transform (push data through mapping functions until done)

-Load (Write to destination)

Pivot vs Unpivot

-Create a “summary table”

-Pivot = “skinny and tall” & Unpivot = “short and wide”

-Usually store unpivoted data—easier to manage

Storing Passwords:

-Hashing is deterministic and noninvertible

Indexing:

-Indexes can be assumed to be already loaded into memory

-index doesn’t have to contain all tuple data (only key values are stored in the index), all tuples = covering index

-indexes are access points for tables

B Tree: (search tree like binary but large numbers of children per node and nodes annotate max values)

-tree/node is memory efficient

-left pointer to values less than entry

-right pointer to values greater/equal to entry

B+ Tree Different:

- leaf nodes point to data, data is searchable by key value annotated by node labels

-leaf nodes (base of tree) form a linked list

Clustered vs Unclustered Index

-Depends on how the actual data is sorted on disk

-Clustered is one that has the same key ordering as what is on the disk(one per table bc stored in same place as table)

-Sequential file, clustered always speed up query but only one per table

-Unclustered index may exist without any ordering on disk, stored in separate place than table (any # per table)

-Sequential file with a different key or heap file(unsorted)

-Unclustered only speed up selections if <1% tuples match

Benefits of B+ Tree:

-Range queries can be fast bc leaf nodes are in linked list (find data 40-85)

**Estimating Amount of Data Read**

-Selectivity factor = proportion of total data needed

Index-Based Selection:

**Leveraging Indexes:**

**-** Create indexes to match expected

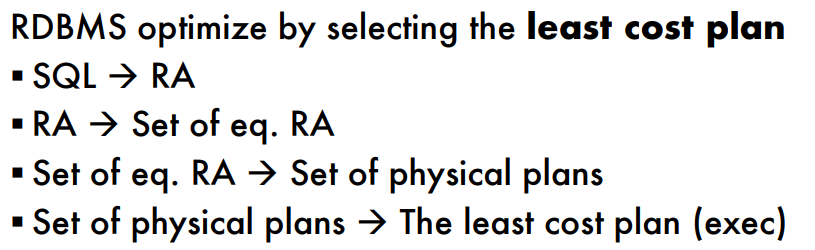
query workload

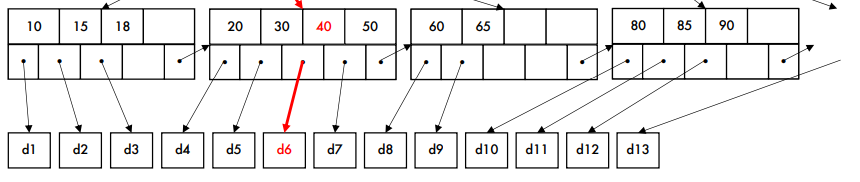
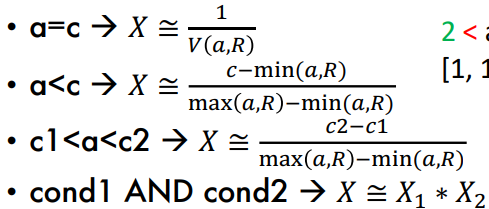
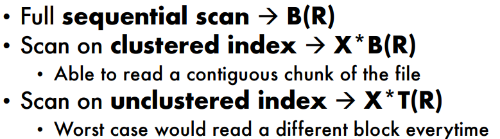
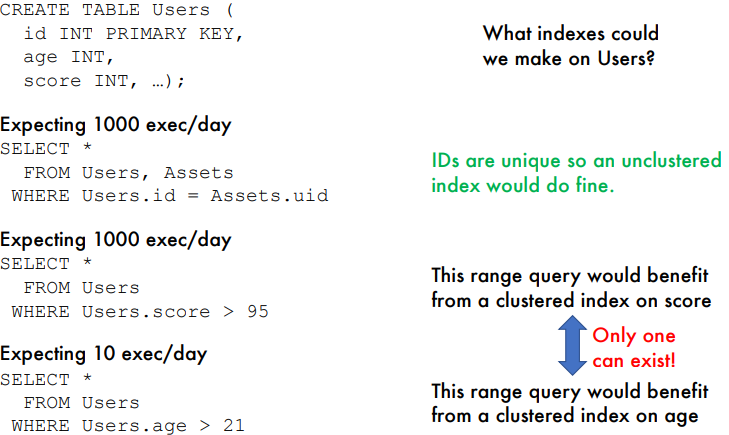
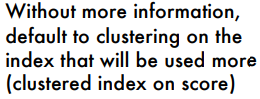
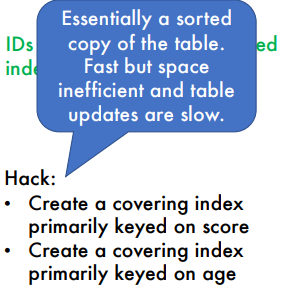
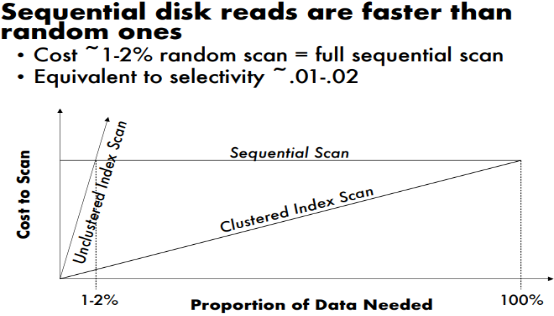
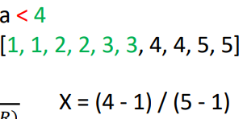
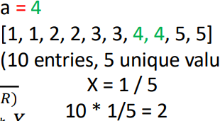
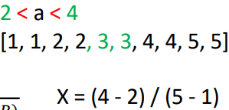
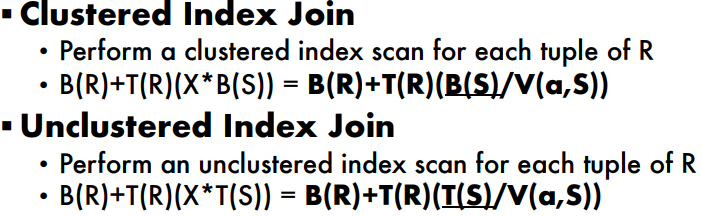
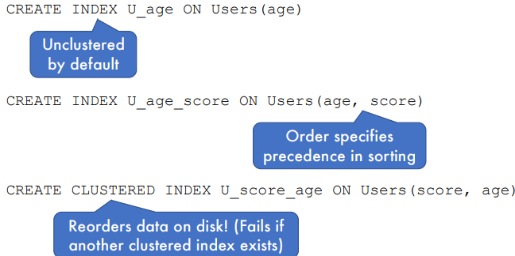
-

Query Cost Estimation

-RA (logical plan) doesn’t describe execution

-RA (physical plan) is with algorithms





Assumptions:

-HDD not SSD

-Row-based storage (tuples stored contiguously)

-IO cost only (reading from disk) only considered

-Cold cache (no data preloaded)

Disk Storage:

-Tables are stored as files

-Heap file -> unsorted files

-Sequential file -> Sorted tuples

RDBMS statistics on our tables:

- B(R) = # of blocks in a relation R

- T(R) = # of tuples in relation R

- V(attr, R) = # of distinct values of attr in R

**Nested Loop Join Algorithm(versatile):**

Equijoin (block at a time):

-Cost = B(R) + B(R) \* B(S)

**Optimized-block Nested Loop Join Algorithm:** (Using multiple tuples in a block and comparing it)

- Cost = B(R) + B(R)/N \* B(S)

**Hash Join (“one pass”, fast, needs at least one input to be small): B(R) < M or B(S) < M (mem)**

- Using hash(x) algorithm, split data in hash table

Cost = B(R) + B(S)

assuming B(R) < M and join with all S

-Same as optimized nested loop with B(R) = N

**-Theta Join = Cross Product**

**-Natural Join = Cross Product w/ no duplicates**

**-Equi-Join = Joining with equalities**

Index-Based Equijoin