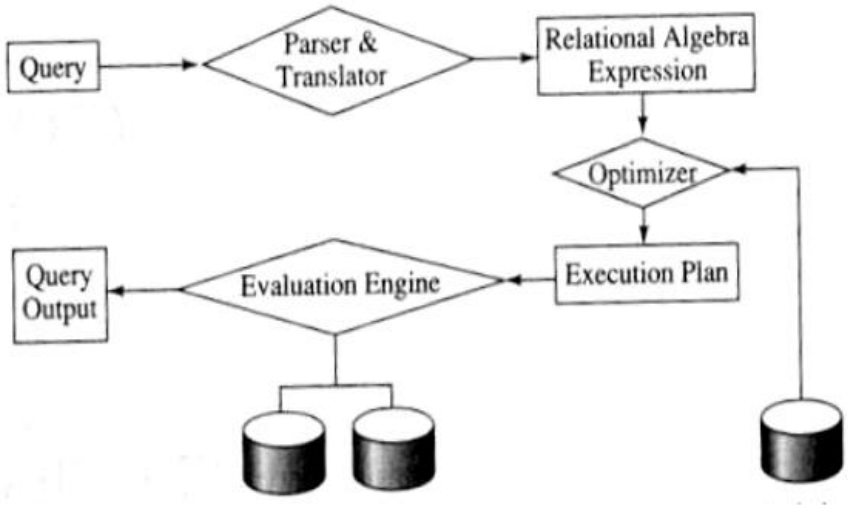
Query Processing

Query processing mainly focuses about what is happening to SQL command inside database engine

Query processing has 3 main steps

- Parsing and translation
  Checking syntaxes and grammar of SQL command
  After checking the syntaxes query will convert into a relational algebra expression
- Optimization
  Produce best execution plan
- Evaluation
  Query output



Parsing and translation

Ex

Select s.sname

From S , SP

Where S.sno = SP.sno AND SP.pno = 'P2'

① $\pi_{s.sname}\left(\sigma_{s.sno=sp.sno\ AND\ sp.pno='P_2'}(S \times SP)\right)$

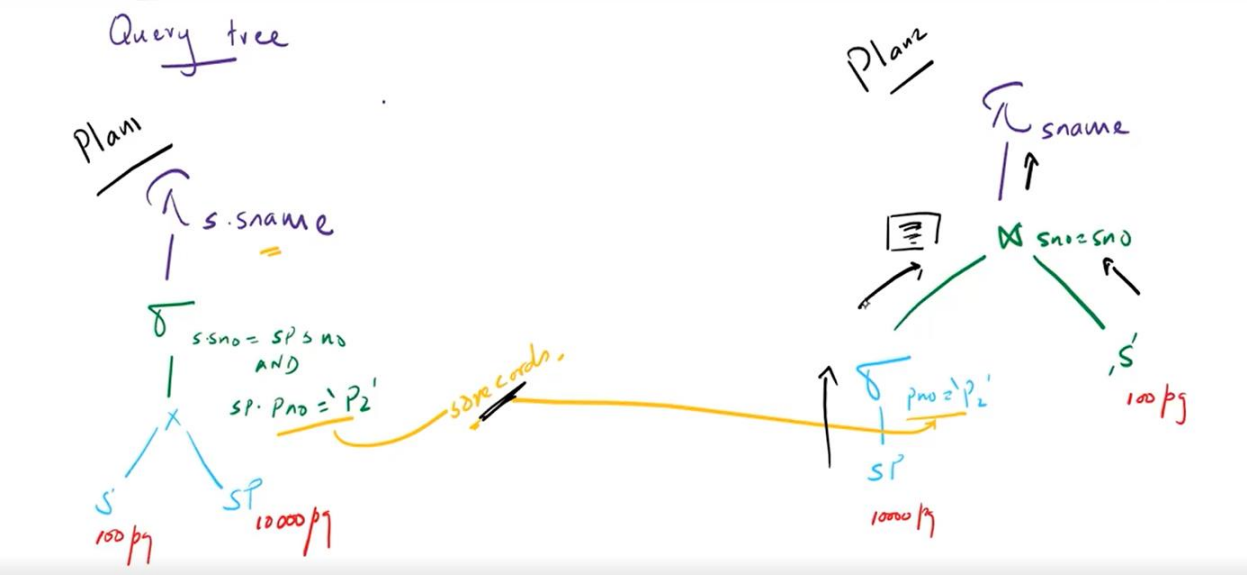
② $\pi_{sname}\left(S \bowtie_{s.sno=sp.sno} \left(\sigma_{pno='P_2'}(SP)\right)\right)$

- Based on I/O cost can decide whether the system will run fast or not
- When have a number of algebra expressions can estimate the cost of algebra expressions to select which one is the best (no of input-output operations)
- Lowest cost is the best execution plan.

Graphical representation of relation algebra expression

Tree structure

plan 1 plan2



Cost = 10000 * 100= 1,000,000 read

1,000,000 written

1,000,000 written

Total cost=3,000,000 I/O

Cost = (10000 +100)

Total cost =10100 I/O

Plan 1

Cartesian product should avoid

Checking condition happen before join

Selection condition we should bring down to query tree

Heuristic Optimization

Heuristic rules are used for algebraic optimization

Minimize plan space

Get few planes

Estimates cost

Query plan

Query plan is a collection of algebra operation

Cost estimation

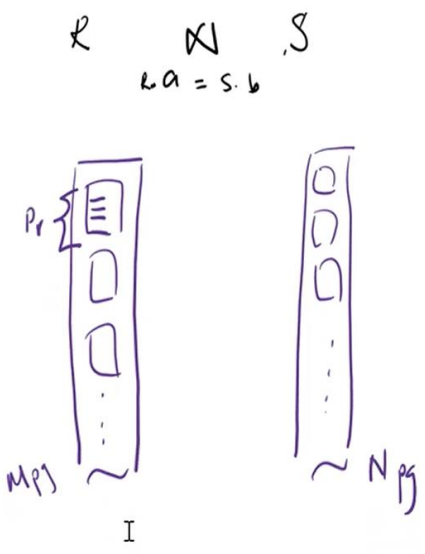
Some algebra operations have many possible ways(algorithm)

Cost may differ based on the chosen algorithm

Joining table algorithm

- Simple nested loop join
- Page oriented nested loop join
- Block nested loop join
- Index nested loop join
- Sort merge join

Checking cost formula



- Simple nested loop join
    - First R table one page take into memory
    - That page each record compared with S table(inner table) each page.
    - This process continuously happen till join all the records of R(when condition is full fill).
    - Consider one record from outer table and that check with inner table.
    - Take outer table record with one by one and compare with inner table.
    - Each outer table record we want to scan inner table record.

Cost SNLJ=M + N*(M *PR)(pages how many time take into computer memory)

M= R table page numbers

N= S table page numbers

PR = number of records in a one page

- Page oriented nested loop join
    - Consider one page of tuples in outer table while reading one page of tuples in inner table.
    - In one iteration join only one page of tuples.

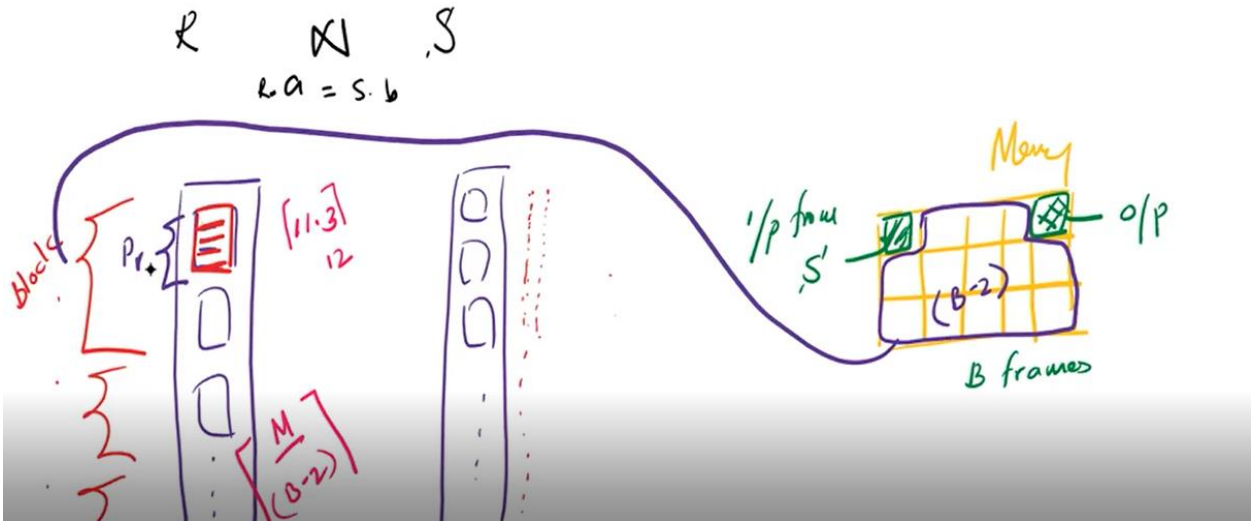
Cost PONLJ=M + N*M

- Block nested loop join

  Consider set of pages
  - All the tuples in a set of pages in outer table while reading all the pages of tuples in inner table.
  - In one iteration join tuples of set pages.

    Cost BNLJ=M + N* # depends of number of blocks

  How to find number of blocks?



  One frame used for S table input also one frame use for S(inner table) table output.
  Remaining frames use for R table(outer table)

  No of data block = pages /B-2

  Cost BNLJ=M + N*(M/(B-2))

- Index nested loop join

  - Create an index on inner table joining column
  - Then take one tuple from the outer table and using index can find exact matching value no need to scan entire table.
  - Each record of R table need to going through the index

  Cost INLJ = M+(index cost)* (M*pr)

  Index file can be B+ tree or hash tree

  Index cost depend on index type

B + tree cost varied (2-4) I/O –--height of B+ tree 2,3,or 4

Extensible Hash cost 1.2 I/O

- Sort merge join

Cost SMJ = (sorting cost) + (Merging cost)

Cost SMJ= Sort R + Sort S + Merging cost

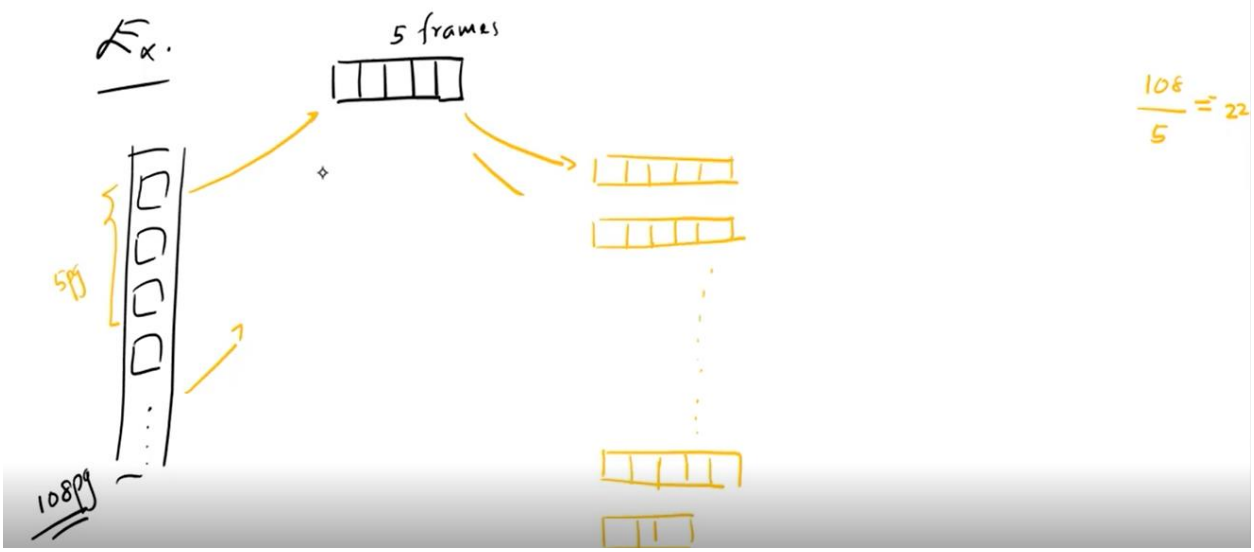
=2M*#no of passes +2N*#no of passes +(M+N)(reading top to bottom of 2 files)

Sorting algorithm

External merge sort algorithm

When handling a database difficult to load all the data set into computer memory for apply sorting algorithm

External merge sort algorithm use to handle above difficulties in database server



Above example, file has 108 pages we want to sort those 108 pages

Memory (buffer) file has 5 frames(size)

The file has 108 pages but all the pages cant load into memory because buffer has only 5 frames(pages).

Want to sort 108 pages using 5 frames.

Algorithm has few(2) passes

Pass 0

- pages load to the memory according to the buffer size and then sorted pages internally and result write into hard disk.
- end of pass 0 we have all sorted files
- according to the above example we get 22 sorted files=108/22

Pass 1

- Merging those sorted files
- B-1 way merging doing during pass 1(B mean number of buffer frames)

  Ex: there are separately sorted 2 files here. need to single sorted files from hose 2 files.
  If the buffer has 3 frame size
  Two frames need to load data from 2 files and one frame to prepare an output.

- First, take one page from 2 files and load to already separated 2 frames then compare values and prepare output finally we can get one merge sorted file
- If have 2 files to sort we need 3 buffer frames. always we need to an extra one to prepare an output.
- When merging need to come top to bottom of sorted files then can compare those values
- End of pass1 we have 6 sorted merge files = 22/(5-1)==6
- End of pass 1 → one file has 20 pages last file has only 8 pages. because at once we merge the b-1 file. so we merge 4 files at a once and one file has 5 pages =one sorted file size= 4*5=20

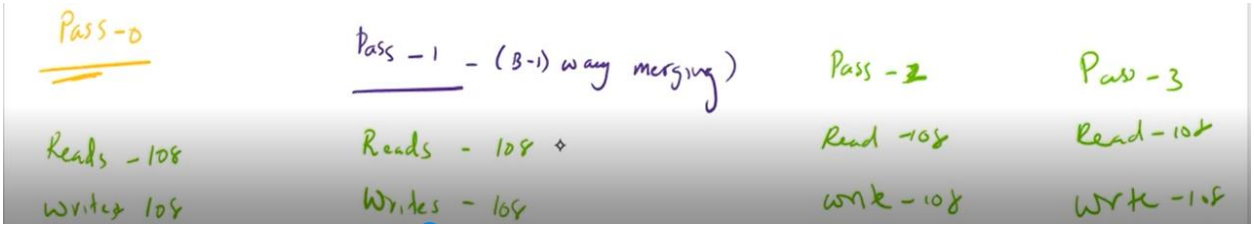
Pass2

- Continuation of merge sorted file into one sorted file.
- End of pass2 we have 2 sorted merge files =6/(5-1)==2
- End of pass 2 → one file has 80 pages the last file has only 28 pages. because at once we merge the b-1 file. so we merge 4 files at a once and one file has 20 pages =one sorted file size= 4*20=80

Pass 3

- Continuation of merge sorted file into one sorted file.
- End of pass3 we have 1sorted merge files =2/(3)==1

Cost calculation



Every pass same data set read into memory and write back to disk

Cost =2 * 108* 4

=864

=2 * no of pages * no of passes

Cost EMS= 2N * passes