Query Processing Basics

- Basic Algorithms
  - Computing Projections
  - Computing Selection
  - Computing Joins

Query Processing Basics

- File Scan
- Sorting
- Indexes

Computing Projection

- Duplicates allowed
  - Scan table keep attributes
  - If F pages in table then F reads + F or less writes
- Duplicates not allowed (Distinct)
  - Sort-based projections
    - Sort and remove duplicates at write of last merge phase
    - Cost same as sorting
  - Hash-based projections
    - Hash into buckets, remove duplicates in each bucket
    - Cost is 4F assume the bucket fits in memory

Computing Selection

- Selection with simple conditions
- $\sigma_{\text{attr op value}} R$
  - No index
    - File Scan
  - B+Tree index
    - Search for B+Tree node where attr = value and scan leaves based on the operator
  - Hash index
    - Only works for attr = value

Computing Selection

- Selection with complex conditions
  - Selections with conjunctive conditions
    - Use the most selective access path
      - Scan the tuples returned by that access path
      - The access path chosen depends on the indexes available
    - Use multiple access paths
      - Use intersection of the tuples returned by all access paths
  - Selections with disjunctive conditions
    - If all disjuncts have better access path than scan, use them otherwise scan

Selection Problem

- Suppose you have a relation R with the following characteristics:
  - 5,000 tuples with 10 tuples per page
  - A 2-level B+tree index on attribute A with up to 100 index entries per page
  - Attribute A is a candidate key of R
  - The values of A are uniformly distributed in the range 1 to 100,000
- a. If the index is unclustered, how many disk accesses are needed to compute the result of $\sigma_{(A => 2000 \text{ AND } A < 6000)}R$?

- b. How many disk accesses are required to compute the result of the query if the index is clustered?

Computing Joins

- Simple nested loops
- Block-nested loops
- Index nested loops

Simple Nested Loop

- $R \bowtie_{A=B} S$
- foreach $t \in R$ do

  foreach $v \in S$ do

  if t.A = v.B then output (t,v)
- Let F_R and F_S be the number of pages in R and S respectively
- Let N_R and N_S be the number of rows in R and S respectively
- Cost is $F_R+N_R*F_S$
  - The order of the loop matters
  - What if $N_R > N_{S?}$
- Cost of output?

Simple Nested Loop Join Problem

- Suppose you have relations R and S with the following characteristics:
  - R has 800 pages with 20 rows per page
  - S has 200 pages with 10 rows per page

- How many disk reads are done to compute $R \bowtie_{R.A = S.B} S$ using a simple nested loop?

Block-nested Loops

- $R \bowtie_{A=B} S$
- foreach page p_r of R do

  foreach page p_s of S do

  output $p_r \bowtie_{A=B} p_s$
- Cost
  - $F_R+F_R*F_s$
- Improvement using more buffer space
  - Assume M page buffers are available
  - Read M-2 page from R and join with a page from S
    - $F_R+F_s*ceiling(F_R/(M-2)$

Block Nested Loop Join Problem

- Suppose you have relations R and S with the following characteristics:
  - R has 800 pages with 20 rows per page
  - S has 200 pages with 10 rows per page
  - Main memory has 52 page buffers

- How many disk reads are done to compute $R \bowtie_{R.A = S.B} S$ using a block nested loop?

Index-nested Loop

- $R \bowtie_{A=B} S$
- foreach $t \in R$

  use the index on B to find all the tuples

  $v \in S$ such that $t.A = v.B$

  output (t,v) for each such v
- Cost examples
  - B+tree (height h) index on B in S
    - $F_R+((h+1)+1) *N_R$

Index Nested Loop Join Problem

- Suppose you have relations R and S with the following characteristics:
  - R has 800 pages with 20 rows per page
  - S has 200 pages with 10 rows per page
  - A height 3 B+tree Index on R.A

- How many disk reads are done to compute $R \bowtie_{R.A = S.B} S$ using a index nested loop?