**Deep Bodra**

**5801 1841**

**DBI Week 6(2/10 - 2/14)**

**Detailed explanation of the topics covered in class**

**2/11/2020**

1. External Sort
   1. Implementation of relational operators can becomes relatively easy if we have relations in sorted order
   2. But, the entire relation may not fit in the memory, so we need to have external sorting algorithms
   3. One of which is TPMMS(Two pass multi way merge sort)
   4. Phase 1: Scan phase
      1. If the memory can hold M blocks then load M blocks
      2. Use an in-memory sort algorithm to sort the records in the block
      3. Write out the sorted blocks (each called a run)
      4. Cost = 1Read+1Write = 2 \* # of blocks in a relation
   5. Phase 2: Merge Phase
      1. Load 1 block from each of the runs
      2. Construct an in-memory priority queue containing the head of the runs
      3. Extract minimum from the queue until a block of records are formed and write it out
      4. Repeat until the queue gets empty
      5. Cost = 2 \* # of blocks in a relation
   6. We may make use of lots of phases for merge, but in practice 2 passes are enough
   7. 4PB of data can be sorted with 2 phases
2. Implementing operators using external sort
   1. Duplicate Elimination
      1. Sort the relation
      2. Scan the relation

If told != tnew

output tnew

told = tnew

tnew = GetNext()

* 1. Group by (with aggregate functions)
     1. Sort the relation
     2. Algo

t=current tuple

If groupold = groupnew

Aggregatei.AddItem(t) i=1,2...n

Else

Aggregatei.Finalize() i=1,2...n

* 1. Set Operators
     1. Sort the relations
     2. For t in R and u in S

|  |  |  |  |
| --- | --- | --- | --- |
|  | R ⋂ S | R ⋃ S | R\S |
| t<u | R++ | Output t  R++ | Output t  R++ |
| t=u | Output t  R++  S++ | Output t  R++  S++ | R++  S++ |
| t>u | S++ | Output t  S++ | S++ |

* 1. Selection
     1. No sort
  2. Projection
     1. No sort
  3. Cross Product
     1. No sort
  4. Join
     1. Sort the relations
     2. Cross product between the group of tuples of R and S for which the join attributes have the same value in R and S

**2/13/2020**

1. External Hash
   1. h(t) = hash function that computes a value that determines the similarity between the tuples based on values of a set of attributes
      1. Phase 1: Split phase
         1. Group the tuples with same hash values together
      2. Phase 2: Merge Phase(in memory algo)
         1. Implementation depends on the types of relational operator under consideration
   2. Implementing operators using external sort
      1. Duplicate elimination
         1. δ(R) = δ(R1) ⋃B δ(R2) ⋃B ...
      2. R ⋂ S (Set Intersection)
         1. R=R1 U R2 …. Rm

S=S1 U S2 … Sn

* + - 1. R ⋂ S = (R1 ⋂ S1) UB (R2 ⋂ S2) ...
      2. Remember? UB is a zero cost operation
  1. Advantages of Hash
     1. SSE instruction compatible
     2. Easier to parallelize
  2. Problems with hash
     1. Very few good hash functions for enormous amount of data
     2. Hash functions may split non uniformly

1. Problems with external sort and hash
   1. A lot of intermediate output files can destroy SSD’s
   2. Solution: Write out these files to spindle drives
2. Nested loop Join(R ⋈ S on A)
   1. It is an alternative way to compute joins for limited amount of data
   2. A dumb strategy is

For t in R

For u in S

If t.A == u.A

Output t.u

Cost = (# of blocks in R) \* (# of tuples in S)

* 1. A little Improvement

For bR in (Blocks of R)

For bs in (Blocks of S)

For t in bR

For u in bs

Output t.u

Cost = (# of blocks in R) \* (# of blocks in S)

* 1. Nested loop method

For bR in (b1, b2, ..bm-2 )

For bs in Bs

For t in bR

For u in bs

Output t.u

Cost =

**TIP TIME:** It is better to have a smaller relation in the outer for loop

--THANK YOU--