CSC 430/530 Database Management Systems / Database Theory

PHYSICAL QUERY PLAN

LECTURE 20

28

Query Processing

- Efficient Query Processing crucial for good or even effective operations of a database
- Query Processing depends on a variety of factors, not everything under the control of DBMS
- Insufficient or incorrect information can result in vastly ineffective plans
- Query Cataloging

29

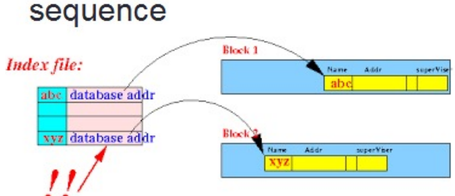
Physical Query Plan Operators

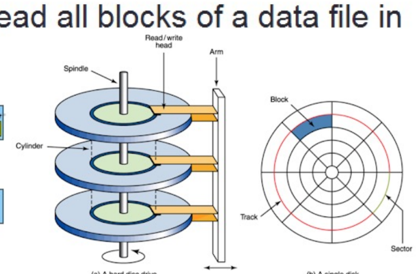
- Basic set of operators that define the language of physical query execution plans
- Comprises the set of relational operators in addition to some more required operators

30

Physical Query Plan Operators

- Table scan operator:
  - Scan and return an entire relation R
  - Scan and return only those tuples of R that satisfy a given predicate
- Table scan: reading all blocks of a sorted data file in sequence
- Index scan: Use an index to read all blocks of a data file in sequence





31

Sorting while Scanning

- Sort-scan operator sorts a relation R while scanning it into memory
  - If sorting is done on an indexed key attributed, no need to do anything; scanning will read R in sorted order
  - If R is small enough to fit in main memory, perform sorting after scanning
  - Else, use external sort-merge techniques to implement sort-scan

32

Iterators

- Iterators are physical-query-plan operators that comprise of three stages:
  - Open() where an iteratable object (such as a relation is opened)
  - GetNext() returns the next element of the object
  - Close() closes control on the object

33

Iterators (Example)

- Example of a "Table scan" iterator:

```
Open() {
        b ← first block of R;
        t ← first tuple of b;
        }

GetNext () {
        if (t is beyond the last tuple in b){
        increment b;
        if (b is beyond last block)RETURN NoMoreData;
        else t ← first block of b;
        }
```

34

Iterators (Example)

```
                oldt ← t;
                 increment t;
                 RETURN oldt;
        }

    Close () {
        }
```

35

Iterators (Example 2)

- Computing Bag union R+S using iterators over iterators

```
        Open (){
        R.Open();
        CurRel← R;
        }

GetNext (){
If (CurRel = R) {
        t ← R.GetNext();
        if (t != NoMoreData)
                return t;
```

36

Iterators (Example 2)

```
else { /* R is exhausted */
        S.Open();
        CurRel ← S;
    }
} ELSE { /* if (CurRel =R ) */
Return S.GetNext;}

Close () { R.Close(); S.Close()}
```

37

Database Access Algorithms

- Algorithms for database access can be broadly divided in the following categories:
  - Sorting-based methods
  - Hash-based methods
  - Index-based methods

38

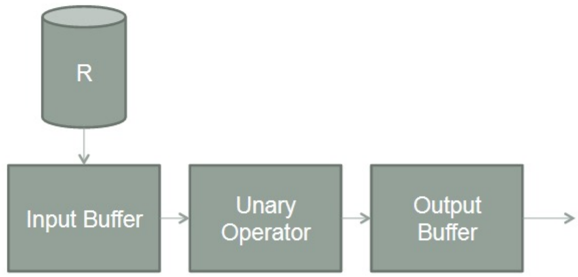
Physical Query Plan Operator Types

- Tuple-at-a-time unary operators:
  - Can read only one block at a time and required to work with only one tuple
- Full-relation unary operators:
  - Requires knowledge of all or most of the relation. Read into memory for small relations in one-pass algorithms
- Full-relation binary operators:
  - Same as above, but on two relations.

39

Tuple-at-a-time Unary Operations

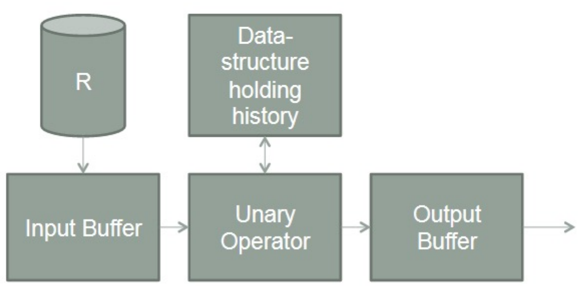
- Examples: σ(R), π(R) etc…
- A strategy for a **one-pass algorithm**:



40

Relation-at-a-time Unary Operations

- Examples: UNIQUE, GROUP BY, etc…
- A strategy for a **one-pass algorithm**:



41

Relation at a time Binary Operators

- One-pass strategies for binary relation-at-a-time operators vary between different operators.
- Almost all of them require at least one of the relation to be completely stored in memory

42

Strategies

- Set Union R U S
- Assuming R is the bigger relation:
  - Read S into memory completely and make it accessible through an in-memory index structure
  - Output all tuples of S while reading
  - For each tuple of R, search if it already exists in S, and output if not.

43

Strategies

- Set Intersection R ∩ S
- Assuming R is the bigger relation:
  - Read S into memory completely and make it accessible through an in-memory index structure
  - For each tuple of R, search if it already exists in S, and output if not.

44

Strategies

- Set Difference R - S
- Assuming R is the bigger relation:
  - Read S into memory completely and make it accessible through an in-memory index structure
  - For each tuple of R, search if it already exists in S.
  - If tuple exists in S, then ignore; else output the tuple.

45

Strategies

- Set Difference S-R
- Assuming R is the bigger relation:
  - Read S into memory completely and make it accessible through an in-memory index structure
  - For each tuple of R, search if it already exists in S, and delete it from S if it exists.
  - Output all remaining tuples of S.

46

Strategies

- Cross Product R x S
- Assuming R is the bigger relation:
  - Read S into memory completely and store it in a buffer. No special data structure required
  - For each tuple of R, combine it with each tuple of S and output the result.

47

Strategies

- Natural Join $R \bowtie S$. Assume R(X,Y) and S(Y,Z)
- Assuming R is the bigger relation:
  - Read S into memory completely and store it in a balanced tree index structure or a hash table.
  - For each tuple of R, search S to see if a matching tuple exists.
  - Output if matching tuple found.

48

One-pass algorithms

- One-pass algorithms are applicable only when one of the relation fits completely into memory.
- In addition, there is enough memory to store at least one block of the other relation
- Hence, if M memory buffers are available, then one of the relations should have a maximum size of M-1.

49

One-pass algorithms

- One-pass algorithms rely on correctly estimating relation sizes and allocating memory buffers.
- If too many buffers are allocated, there is a possibility of **thrashing**
- If too few buffers are allocated, then one-pass algorithms may not run

50

Summary

- Stages in Query Processing
- Logical Query Plan and Physical Query Plan
- Intermediate Query Language
- Physical Query Plan Language constructs
- One-pass algorithm for unary and binary operators.

51

Multi-pass Algorithms

- Used when entire relations cannot be read into memory
- Requires alternate computation and retrieval of intermediate results
- Many multi-pass algorithms are generalizations of their corresponding two pass algorithms

52

Basic idea: Two-pass Algorithms based on Sorting

- Suppose relation R is too big to fit in memory which can accommodate only M blocks of data.

- The “sorting-based” 2-pass algorithms have the following basic structure:
  1. Read M blocks of records into memory and sort them
  2. Write them back to disk
  3. Continue steps 1 and 2 until R is exhausted
  4. Use a variety of “query-merge” techniques to extract relevant results from all the sorted M-blocks on disk

53

Example of two-phase algorithm:

- Query: Duplicate elimination using sorting

- Phase 1:
  - Let relation R, in which duplicates must be eliminated, be too big to fit in memory.
  - **Step A:** Read M << R blocks into memory and sort them.
  - Store the sorted set of M blocks back on disk
  - Return to step A till R is exhausted.

54

Example of two-phase algorithm:

- Phase 2:
  - Take one block from each sorted sub-list on disk and eliminate duplicate tuples
    - **More specifically:** Take the first element p of the first block and move all other blocks to go beyond p.
  - Since blocks are sorted, the "merge-elimination" take $O(M \times n \times b)$, where M is the number of blocks in a sorted block set, n is the number of block sets, and b is the number of tuples in a block.

55

Duplicate elimination using sorting

	In Memory	Waiting on Disk	
1	1, 2	2, 2	2, 5
2	2, 3	4, 4	4, 5
3	1, 1	2, 3	5

Output 1 from the first block and move all blocks beyond "1"

	In Memory	Waiting on Disk	
1	2	2, 2	2, 5
2	2, 3	4, 4	4, 5
3	2, 3	5	

56

Duplicate elimination using sorting

	In Memory	Waiting on Disk	
1	2	2, 2	2, 5
2	2, 3	4, 4	4, 5
3	2, 3	5	

Output 2 from the first block and move all blocks beyond "2"

	In Memory	Waiting on Disk	
1	5		
2	3	4, 4	4, 5
3	3	5	

57

Example of two phase algorithm: (2)

Set Union using sorting is simply duplicate elimination from two sets.

For computing R U S:

1. Read blocks from R and S, sort them and store them on disk.
2. Repeat step 1 until R and S are exhausted.
3. Use duplicate elimination over the set of all these blocks outlined in the previous example.

58

Example of two phase algorithm: (3)

Set intersection using sort:

1. Given two relations R and S, read them in terms of blocks and store the blocks on disk in sorted order
2. Take each block of R stored on disk and perform the following:
   1. Read the first tuple t from R's block
   2. Move all blocks of S beyond t (i.e., travers until t in S)
   3. If t existed in any block of S, output t, else ignore and move onto the next tuple in R.
3. (Note: Set intersection assumes R and S are sets; i.e. no duplicates)

59

Example of two phase algorithm: (4)

Natural Join Using Sort

In order to compute $R(X,Y) \bowtie S(Y,Z)$ using sorting:

1. Read blocks of R and sort them on the Y attribute and store them back on disk
2. Read blocks of S and sort them on the Y attribute and store them back on disk
3. For each value y at the top of the first block of R:
   1. If y occurs in S, then read all blocks of R beyond y and store them separately

60

Summary

- Multi pass algorithm
  - We know the two-phase algorithm
  - Duplicate elimination is important
  - Review basic operators

61