Data Base Management Systems

Query Processing

Dr. Wenceslao PALMA
wenceslao.palma@ucv.cl

March 2011

We will take a look to traditional query processing before examinate query operators in the context of DSMS.

Traditional Query Processing

Query Processor

The QP is the group the components of a DBMS that turns user queries and data-modification commands into a sequence of operations on the database and executes those operations. There are three broads steps that the query processor must take:

- The query is *parsed*, that is, turned into a parse tree representing the structure of the query in a useful way.
- The parse tree is converted to an initial query plan (*Query rewrite*) referred as *logical query plan* which is expected to require less time to execute.
- The *logical query plan* must be turned into a *physical query plan*, which indicates not only the operations performed, but the order in which they are performed,the algorithms used to perform each step, and the ways in which the data is obtained and data is passed from one operation to another.

Traditional Query Processing

Query Processor

To select the best *query plan* the query processor must decide:

- Which of the algebraically equivalents forms of a query leads to the most efficient algorithm for answering the query.?
- What algorithm should we use to implement each operation?
- How the operations pass data from one to the other? pipelined fashion, main memory buffers or via the disk.

Traditional Query Processing

Physical Query Plan Operators

- Physical query plans are built from operators, each of which implements one step of the plan.
- The physical operators are particular implementations for one of the operators of relational algebra.
- However, there are physical operators for other tasks that do not involve an operator of relational algebra, e.g., *table scan* and *index scan*

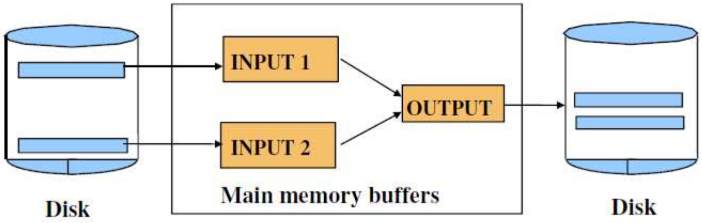
Traditional Query Processing

Physical Query Plan Operators-*sort-scan*

- A query could include an **ORDER BY** clause.
- The operator *sort-scan* takes a relation R and a specification of the attributes on which the sort is to be made, and produces R in that sorted order. There are several ways to implement *sort-scan*:
    - If R is stored as an sorted indexed sequential file $\rightarrow$ scan the index.
    - If R is small enough to fit in main memory $\rightarrow$ main memory sorting algorithm.
    - If R is too large to fit in main memory $\rightarrow$ multiway merging algorithm.

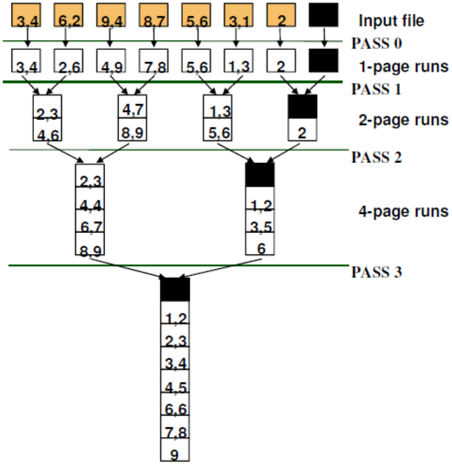
Traditional Query Processing- External Sorting

When a collection of records is too large to fit in main memory, the only practical way to sort it is to read some records from disk, do some rearranging, then write them back to disk. This process is repeated until the file is sorted, with each record read perhaps many times
In a two-way external mergesort we need 3 main memory buffers.

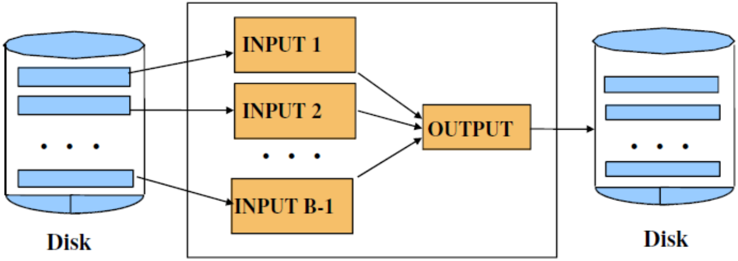


Traditional Query Processing- External Sorting

- Each pass we read/write the N pages of the file.
- The number of steps is $1 + log_2 N$.
- Total cost of I/Os is $2N(1 + log_2 N)$, and the order of magnitude is $O(N log_2 N)$.
- What if we have B buffers?



Traditional Query Processing- External Sorting



- Each pass we read/write the N pages of the file.
- In the first pass we write N/B runs.
- The number of steps is $1 + log_{B-1} N/B$.
- Total cost of I/Os is $2N(1 + log_{B-1} N/B)$, and the order of magnitude is $O(N log_{B-1} N)$.

Traditional Query Processing- External Sorting

Problem

Suppose we have a file of 10000 pages and a disk with an average seek time of 10ms, average rotational delay of 5ms, and a transfer time of 1ms for a page. You are given 64 buffer pages and asked to sort the 10000 pages file. Find the cost of the following operations:

- Case 1: 63 input buffers of 1 page each and 1 output buffer page.
- Case 2: 3 input buffers of 16 pages each and 1 output buffer of 16 pages.
- Case 3: 13 input buffers of 4 pages each and 1 ouput buffer of 12 pages.

Traditional Query Processing

Iterators

- Conceptually, a query operator can be thought of as a function that consumes input data, stores some state, performs computation, modifies the state and output results.
- Using iterators tuples pass between operators as needed. Thus, tuples produced by one operator are passed directly to the next operators that uses it, whitout ever storing the intermediate tuples on disk. This approach is called *pipelining*.
- An iterator is implemented overloading the following methods: Open(), GetNext() and Close().

Traditional Query Processing

The Join Operator - Tuple-Based Nested-Loop Join Alg.

```
For each tuple s in S do
    For each tuple r in R do
        if r and s join to make a tuple t then
            output t;
```

The Join Operator - Iterator for Tuple-Based Nested-Loop Join

```
Open(R,S){
    R.Open();
    S.Open();
}

GetNext(R,S){
     REPEAT{
          r:= R.GetNext();
          IF (NOT Found){
              R.Close();
              s:=S.GetNext();
              IF (NOT Found) Return;
              R.Open();
              r:=R.getNext();
          }
     }UNTIL (r and s join);
     RETURN the join of r and s;
}
```

Traditional Query Processing

The Join Operator - Iterator for Tuple-Based Nested-Loop Join

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

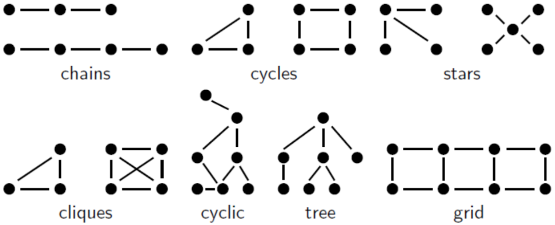
Traditional Query Processing-The Join Ordering Problem

Join ordering

- We know that $A \bowtie B = B \bowtie A$ but the cost is often different
- Objective: choose order which minimizes the sum of the sizes of intermediate results.
- Naïve approach: n! combinations.
- As the number of joins increases, the number of alternative plans grows rapidly; we need to restrict the search space.
- Finding the optimal join ordering is an NP-hard problem.

Traditional Query Processing-The Join Ordering Problem

Queries can be characterized by their query graph [1]



[1] Extracted from Building Query Compilers, Guido Moerkotte

Traditional Query Processing-The Join Ordering Problem

- A join tree is a binary tree where:
  - leaf nodes are the relations
  - inner nodes are joins (and possibly cross products).
  - The edges represent the input/output relationship.
- The most important classes of join trees are: left-deep trees, right-deep trees, zig-zag trees, and bushy trees.
- Choosing the shape of the join tree is equivalent to parenthesize n-way joins.

How we can calculate the number of ways of parenthesize n-way joins.?

Traditional Query Processing-The Join Ordering Problem

Queries considered

- Conjunctive queries: its **where** clause is composed of a conjunction of simple predicates.
- A simple predicate is of the form $\epsilon_1 \theta \epsilon_2$, where $\theta \in \{=, \neq, >, <, \geq, \leq\}$
- Generally, the algorithms consider predicates of the form $A = B$, such joins are called equi-joins.

Traditional Query Processing-The Join Ordering Problem

Cost function

- A cost function is used in order to judge the quality of join trees.
- Join ordering finds among all equivalent join trees the join tree with lowest associated costs.
- Intermediate results can be estimated using

$$f_{i,j} = \frac{|R_i \bowtie_{p_{i,j}} R_j|}{|R_i| \times |R_j|}$$

- $f_{i,j}$ represents the percentage of tuples in the Cartesian Product that "survive" the predicate $p_{i,j}$
- $f_{i,j}$ is assumed to be given as input to the join ordering algorithm.

Traditional Query Processing-The Join Ordering Problem

We can obtain the cardinality estimation of a join tree applying:

$$|T| = \begin{cases} |R_i| & \text{if } T \text{ is a leaf } R_i \\ (\prod_{R_i \in T_1, R_j \in T_2} f_{i,j})|T_1||T_2| & \text{if } T = T_1 \bowtie T_2 \end{cases}$$

Given a join tree T, the cost function C_{out} is defined as:

$$C_{out}(T) = \begin{cases} 0 & \text{if } T \text{ is a single relation} \\ |T| + C_{out}(T_1) + C_{out}(T_2) & \text{if } T = T_1 \bowtie T_2 \end{cases}$$

Considering: $|R_1| = 10$,$|R_2| = 100$,$|R_3| = 1000$,$f_{1,2} = 0.1$,$f_{2,3} = 0.2$ and assume $f_{i,j} = 1$ for all other combinations.

(a) $(R_1 \bowtie R_2) \bowtie R_3$?

(b) $(R_2 \bowtie R_3) \bowtie R_1$?

(c) $(R_1 \bowtie R_3) \bowtie R_2$?

Traditional Query Processing-The Join Ordering Problem

- The optimal join ordering can be found by means of dynamic programming when the number of relations is small (the worst case running time is $O(2^N)$).
- Otherwise, in order to provide aceptable join orderings heuristic and randomized algorithms can be used.