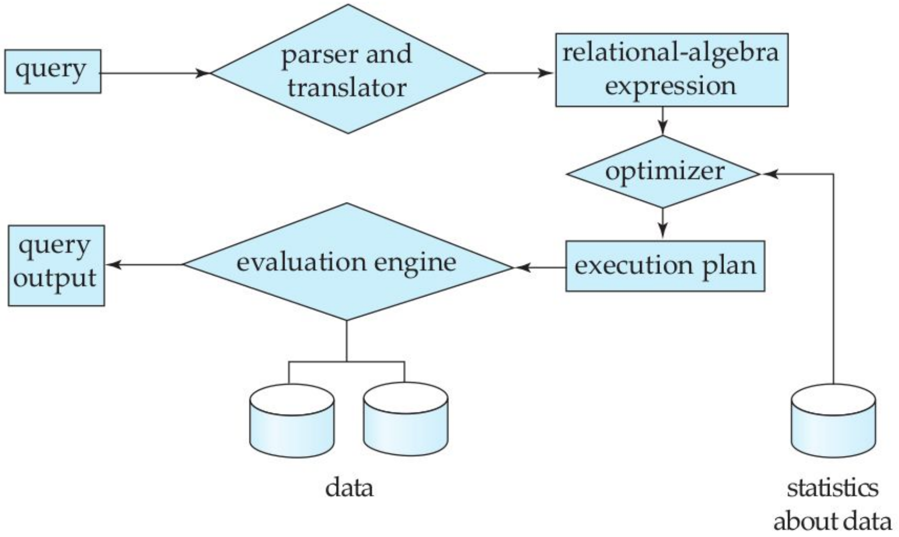
Databases and Information Systems
CS303

Query Processing

20-10-2023

Query Processing

- Mainly 3 steps:
    - Parsing
    - Optimization
    - Evaluation

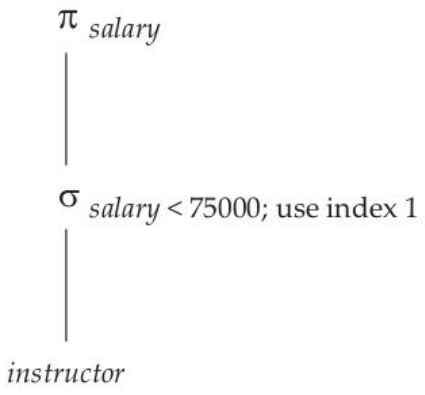


Parser

- Similar to Parser of a Compiler
- Constructs a parse-tree representation of the query,
  - It then translates into a relational-algebra expression
- Verifies Syntax of the query
  - Does the relation names appearing in the query are names of the relations in the database
  - Does the attributes of Select and Where appear in the relations mentioned in the query
- We will not discuss the parsing phase in detail

Relational Algebra Representation

- Same query can have multiple relational algebraic representations
  - SELECT salary FROM instructor WHERE salary < 75000
- $\sigma_{salary < 75000}(\pi_{salary}(instructor))$
- $\pi_{salary}(\sigma_{salary < 75000}(instructor))$
- Even with same relational algebra expression we can use different evaluation schemes
  - Search every tuple in instructor with salary < 75000
  - If B^+ tree is available on salary as search key, we can use index search to locate the tuples

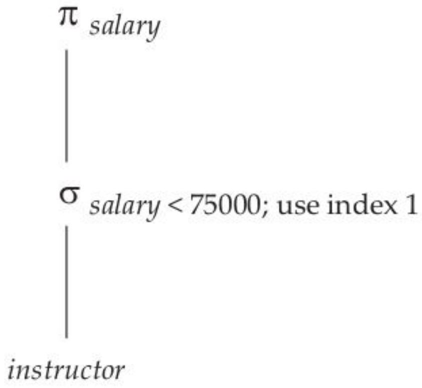


Relational Algebra Representation

- A relational algebra operation annotated with instructions on how to evaluate it is called an evaluation primitive.

- A sequence of primitive operations that can be used to evaluate a query is a query-execution plan or query-evaluation plan

- Query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query.



Relational Algebra Representation

- Different evaluation plans have different cost
- Users would not worry about this when writing the query
- System should construct a query-evaluation plan that minimizes the cost of query evaluation
    - This task is called query optimization
- Instead of using the relational-algebra representation:
    - several databases use an annotated parse-tree representation based on the structure of the given SQL query.

Relational Algebra Representation

- In order to optimize a query, a query optimizer must know the cost of each operation.
    - Although the exact cost is hard to compute, since it depends on many parameters
    - Possible to get a rough estimate of execution cost for each operation

- How is the cost estimated for a given execution plan ?

Measures of Query Cost

- Estimate cost of individual operation and combine them to get the cost of the query

- Cost is measured in terms of:
    - Disk accesses
    - CPU time to execute a query
    - In a distributed or parallel database system, the cost of communication

- In large databases, cost to access data from disk is the most important cost

- Let us ignore CPU time cost and only study disk access cost
    - Actual implementations consider both

Measures of Query Cost

- Use the number of block transfers from disk and the time to access the block to estimate the cost of a query-evaluation plan

- If it takes an average of t_T seconds to transfer a block of data, and has an average block-access time (disk seek time plus rotational latency) of t_S seconds
    - Transferring b blocks and performs S seeks would take $b * t_T + S * t_S$ seconds
    - Typically t_T = 0.1 milliseconds and t_S = 4 milliseconds
        - With 4KB block size, this gives transfer rate of 40MB per second

- Some database systems perform test seeks and block transfers to estimate average seek and block transfer costs, as part of the software installation process

- These costs do not include writing the output to the disk
    - Taken into account if required

Measures of Query Cost

- The costs of the algorithm depends on the size of the buffer in main memory.
  - In the best case, all data can be read into the buffers, and the disk does not need to be accessed again.
  - In the worst case, the buffer can hold only a few blocks of data
    - Assume one block per relation

- When presenting cost estimates, we assume the worst case

- We assume that data must be read from disk initially
  - It is possible that a block that is accessed is already present in the in-memory buffer.
  - For simplicity, we ignore this effect
  - The actual disk-access cost during the execution of a plan may be less than the estimated cost.

Measures of Query Cost

- Response time of a plan is very hard to estimate without actually executing the plan, for the following reasons:
    - Depends on the contents of the buffer when the query begins execution
    - In a system with multiple disks, the response time depends on how accesses are distributed among disks
        - Needs detailed knowledge of data layout on disk.
- A plan may get a better response time at the cost of extra resource consumption.
    - If a system has multiple disks, a plan A that requires extra disk reads, but performs the reads in parallel across multiple disks may finish faster than another plan B that has fewer disk reads, but from only one disk
    - But if many instances of a query using plan A run concurrently, the overall response time may be more than if the same instances are executed using plan B
        - since plan A generates more load on the disks
- Instead of trying to minimize the response time, optimizers generally try to minimize the total resource consumption of a query plan

Cost estimation for SELECTION OF TUPLES

- File-scan is the lowest level operator to access data
    - Search algorithms that locate and retrieve records that fulfill a selection condition
    - File-scan allows an entire relation to be read in cases where the relation is stored in a single, dedicated file.
    - Assume that a records of a relation is stored in continuous blocks

Cost estimation for SELECTION OF TUPLES

- Linear Scan : Scan each file block and test all records to see whether they satisfy the selection condition.
  - An initial seek is required to access the first block of the file
  - Can be applied to any file, regardless of the ordering of the file, or the availability of indices, or the nature of the selection operation.

  - Cost : $t_S + b_r * t_T$
    - Where b_r is the number of blocks used by the file

- Scan based on index structures (Index Scan):
  - A primary index (clustering index) is an index that allows the records of a file to be read in an order that corresponds to the physical order in the file (usually primary key)

  - An index that is not a primary index is called a secondary index

Cost estimation for SELECTION OF TUPLES

- Various possibilities on search:
  - Primary index is a key (unique) and test condition is for equality of search-key : retrieve the single record that satisfies the equality
  - Primary index is a non-key and test condition is for equality of search-key: retrieve multiple records that satisfy non-key equality
  - Secondary index, test for equality of search-key: One or multiple records depending on search index is key or not
  - Primary index, test for Comparison :
    - For A > v or A >= v; find the first record with value equal A equal to v; Output everything after that
    - For A <= v or A < v; index lookup is not required. Start from the beginning of the file, continue upto the value v
  - Secondary index, test for Comparison :
    - Use the index structure to get pointers to location of the records

	Algorithm	Cost	Reason
A1	Linear Search	$t_S + b_r * t_T$	One initial seek plus b_r block transfers, where b_r denotes the number of blocks in the file.
A1	Linear Search, Equality on Key	Average case $t_S + (b_r/2) * t_T$	Since at most one record satisfies condition, scan can be terminated as soon as the required record is found. In the worst case, b_r blocks transfers are still required.
A2	Primary B^+-tree Index, Equality on Key	$(h_i + 1) * (t_T + t_S)$	(Where h_i denotes the height of the index.) Index lookup traverses the height of the tree plus one I/O to fetch the record; each of these I/O operations requires a seek and a block transfer.
A3	Primary B^+-tree Index, Equality on Nonkey	$h_i * (t_T + t_S) + b * t_T$	One seek for each level of the tree, one seek for the first block. Here b is the number of blocks containing records with the specified search key, all of which are read. These blocks are leaf blocks assumed to be stored sequentially (since it is a primary index) and don't require additional seeks.
A4	Secondary B^+-tree Index, Equality on Key	$(h_i + 1) * (t_T + t_S)$	This case is similar to primary index.
A4	Secondary B^+-tree Index, Equality on Nonkey	$(h_i + n) * (t_T + t_S)$	(Where n is the number of records fetched.) Here, cost of index traversal is the same as for A3, but each record may be on a different block, requiring a seek per record. Cost is potentially very high if n is large.
A5	Primary B^+-tree Index, Comparison	$h_i * (t_T + t_S) + b * t_T$	Identical to the case of A3, equality on nonkey.
A6	Secondary B^+-tree Index, Comparison	$(h_i + n) * (t_T + t_S)$	Identical to the case of A4, equality on nonkey.

Cost estimation for SELECTION OF TUPLES

- Complex comparison conditions:
  - Conjunctive selection : $\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n}(r)$
    - Conjunctive selection using one index
      - If any θ_i is a simple condition on some search-index, retrieve all records that satisfy θ_i (using one of the previous algorithms) and check which of these records satisfy all other conditions.
        - $\sigma_{\theta i}(r)$
      - To reduce the cost, choose one of the previous search algorithm that gives best result for the chosen θ_i
      - Cost is same as the cost of the chosen algorithm

Cost estimation for SELECTION OF TUPLES

- Complex comparison conditions:
  - Conjunctive selection : $\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \ \theta n}(r)$
    - Conjunctive selection by intersection of identifiers:
      - Compute the pointers to records satisfying each θ_i (where index-search is available)
      - Take the intersection of all the resulting pointers
      - Retrieve the records and check for conditions for which index-search is not available

      - Cost is sum of the cost of each individual index-scans
        - Can be improved by sorting the pointers and retrieving records in other
          - Every block is retrieved at most once