Access Path Selection in Relational Database Management Systems

--- G. Selinger, M. Astrahan, D. Chamberlin, A. Lorie, G. Price

Summary

The main problems addressed in this paper are

- How system R chooses access paths for both simple queries and complex queries, given an expression.
- Issues of access path selection for queries and the role of optimizer in processing SQL query.
- Storage component access path available in physical table and the costs associated with optimizer for single table queries.
- The join operation between two or more tables and costs associated with it and Processing Nested Queries.

It is Important to make application programs and user activities independent on growth in data type and data representation because the applications may only work for a particular representation of data and may fail if there is a change in data representation. It is also important that the data should be consistent when there is a change in data representation.

The Main Ideas discussed are

- **Processing of SQL Statement**: Contains four phases: parsing, optimization, code generation, and execution. Parser checks the syntax of the query. Optimizer looks for referenced tables in the database and verifies the existence. Checks for semantic and data type errors and finally performs the Access path selection. Code Generator translates Access Specification Language into machine language code to execute the code. Then the code is executed and calls the internal storage system interface to scan the relations in the query.
- **Research Storage Systems (RSS)**: Storage subsystem of System R. Maintains physical storage of relations, access paths, locking, logging &

Recovery facilities. Two types of RSS scans for accessing tuples: Segment Scan & Index Scan. Segment scans all tuples in a relation and returns those tuples belonging to given query. Index Scan creates index for one or more columns in a relation by implementing B-Trees. So, The Index scan reads along the leaf nodes and obtain tuple identifiers using which returns the data tuples. Both scans take a set of predicates and if any tuple satisfies the predicate, It is returned. Sargable predicate which uses column comparison operator value.

- **Costs for Single Relation Access Paths**: The optimizer examines predicates in the query and access paths available on relations, and formulates cost prediction for each access path based on I/O Operation and CPU Usage. The each predicate in WHERE condition is known as Boolean Factor, which is matched with index key. The access path using the index column is efficient, when Boolean factor matches with the index. The statistics on relations are maintained in System R catalog. Using these statistics, optimizer assigns a selectivity factor for each Boolean factor, which gives the fraction of tuples that satisfy the predicate. Choosing an optimal access path consists of using these selectivity factors together with statistics on available access paths.

- **Access Path Selection for Joins**: System R introduced the dynamic programming approach for doing a sequence of joins, telling how to organize the search for the best plan in terms of finding the best join order for successively larger subsets of tables. A heuristic based on join predicates is used to reduce the permutations, which are considered. Only two types of joins are considered: Nested Loop: scan on the outer relation and the first tuple is retrieved; For each outer relation, the inner relation is scanned to retrieve tuples which satisfy the join predicative. Merging scans: require the outer and inner relations to be scanned in join column order. It can takes advantages of the ordering on join columns to avoid rescanning the entire inner relation for each tuple of the outer relation. System R also considers "interesting orders", meaning that at each stage in the algorithm, in addition to the lowest-cost option, it also identifies and considers options that lead to tuple orderings, which could reduce the cost of later stages in the execution plan.
- **Nested Queries**: The sub queries may return a single value or a set of values based on the scalar operators. The sub query needs to be evaluated only once before the main query is executed. If a single value is returned by sub query, it is incorporated in the main query. If sub query returns set of values, They

are returned in a temporary list and the predicate is evaluated similar to the way in which it evaluates for main query. In case of nested sub queries, The deeply nested sub query is executed first before its parent. The sub query containing reference to candidate tuple of higher-level query is called as correlation sub query. This query must be re evaluated for each candidate tuple from referenced query.

The final results are

- The access path selection for single table queries, joins and nested queries are explained by use of statistics, CPU Utilization, and methods of determining join order.
- They give good formulas for estimating the cost of various query predicates based on limited statistics about the indexes and relations, and a robust method for combining that information together as you consider different potential join orders.
- Interesting Orderings for joins avoids the additional work of storage and sorting of intermediate results. The Evaluation techniques for comparing choice of access paths are not mentioned and will be discussed in future papers.

Question: I have a doubt on how optimizer executes correlated sub query.

In paper, they mentioned, "This re-evaluation must be done before the correlation sub query's parent predicate in the higher level block can be tested for acceptance or rejection of the candidate tuple." Can you explain this statement?

Submitted By

Lakshman Madhav Kollipara

SID- 932655504

Submitted To

Arash Termehchy

CS540- Database Management Systems