Query Optimization

Gaurav Wadkar

Department of Computer Science
S.P College
Pune 411029

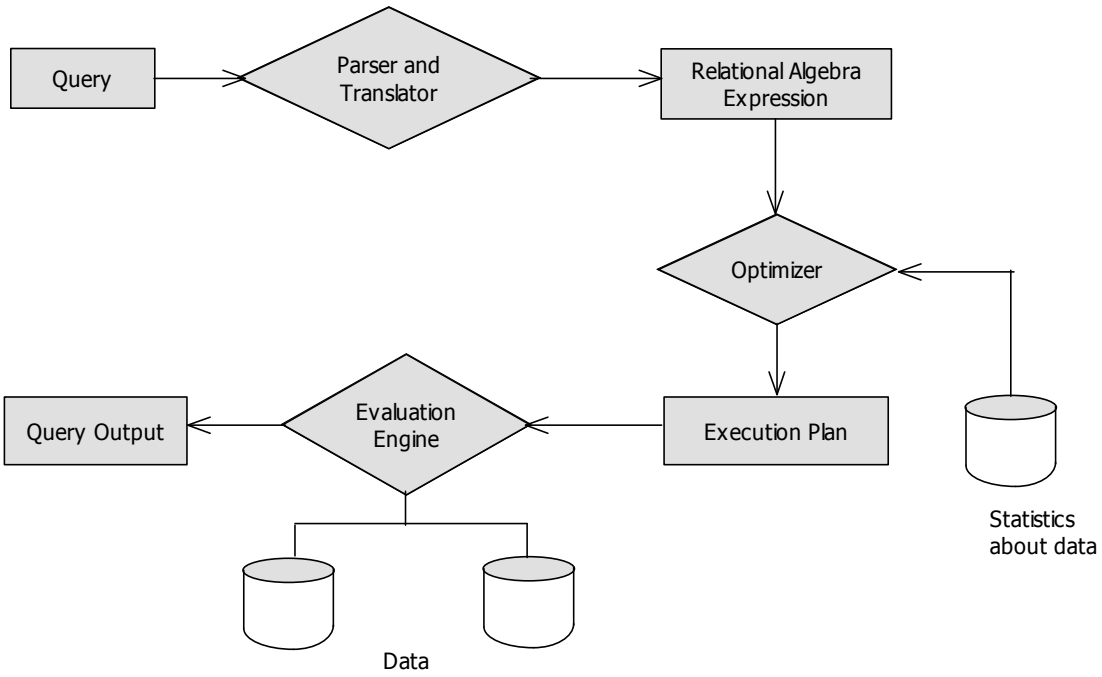
Abstract

The Difference in costs between a good strategy and bad strategy is often substantial and may be several orders of magnitude. This paper describes the process of generating minimum cost queries In order to scrutinize the procedure I have also generated an evaluation plan.
I also describe different tuning strategies and different types of optimizers. All the operations are them compared to the Oracle Query Optimizer for better understanding of the subject.

Introduction

Query Optimization is the process of selecting the most efficient query evaluation plan among the many strategies usually possible for processing a given query especially if the query is complex. The Difference in costs between a good strategy and bad strategy is often substantial and may be several orders of magnitude.

Theory



Structure

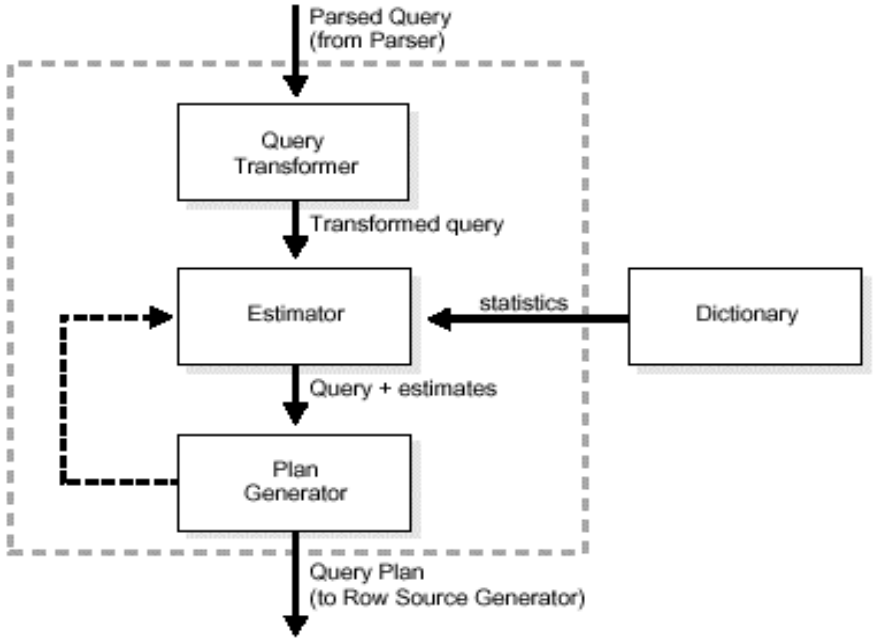
It comprises of the following phases:
Parser and Translator: It reads the SQL Query and translates it to Relational Algebra Expression (Equivalent expression).
Cost Generator: It will generate the cost for all the Equivalent forms of the Query.
Evaluation Engine: It generates Evaluation Plan for the least cost Query.
Error Checker: Checks for any parse and semantic errors.
Optimizer: It generates Equivalent forms of the Query.

Inputs would be in the form of an SQL Expression

In the process some intermediate results in the form of Equivalent Relational Algebra Expressions, Costs for Equivalent Relational Algebra Expressions
And Minimum Cost Equivalent Expression would be generated.

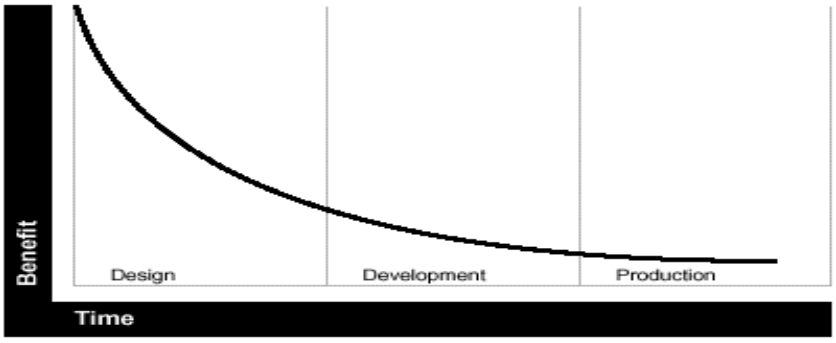
The DBA would be shown an output comprising of an Input Relational Algebra Expression, Optimized Relational Algebra Expression, Costs of Optimized Relational Algebra Expression and Evaluation Plan for Optimized Relational Algebra Expression.

Process



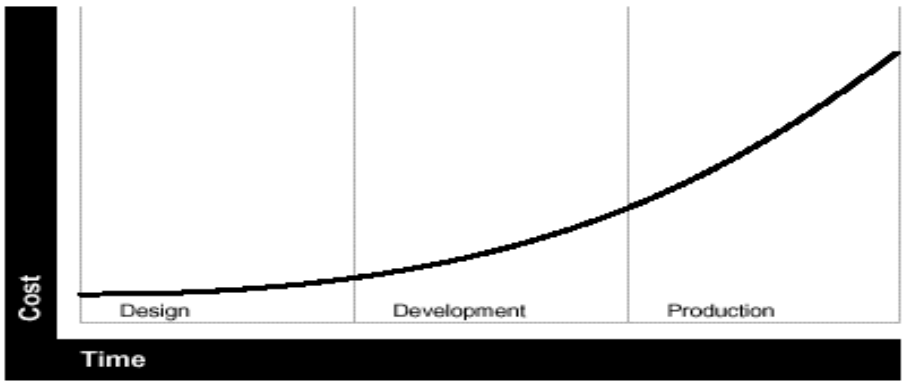
Tuning

Proactive Tuning While Designing and Developing Systems



The most effective time to tune is during the design phase: you get the maximum benefit for the lowest cost.

Reactive Tuning to Improve Production Systems



Prioritized Tuning Steps

- Step 1: Tune the Business Rules
- Step 2: Tune the Data Design
- Step 3: Tune the Application Design
- Step 4: Tune the Logical Structure of the Database
- Step 5: Tune Database Operations
- Step 6: Tune the Access Paths
- Step7:TuneMemoryAllocation
- Step 8: Tune I/O and Physical Structure
- Step 9: Tune Resource Contention
- Step 10: Tune the Underlying Platform

Overview of Data Access Methods to enhance performance

- Using Indexes**--**create indexes on tables that are queried for less than 2% or 4% of the table's rows.
- Consider indexing keys that are frequently used in WHERE clauses.
- Consider indexing keys that are frequently used to join tables in SQL statements
- Do not index columns that are frequently modified.

- Do not index keys that appear only in WHERE clauses with functions or operators.
- Consider indexing foreign keys of referential integrity constraints in cases in which a large number of concurrent INSERT, UPDATE, and DELETE statements access the parent and child tables.
- Consider creating a composite index on keys that are frequently used together in WHERE clause conditions combined with AND operators
- If several queries select the same set of keys based on one or more key values ,then consider creating a composite index containing all of these keys.
- If all keys are used in WHERE clauses equally often, then ordering these keys from most selective to least selective in the CREATE INDEX statement best improves query performance.

Cost Based Optimizer

Access Paths for the CBO

- One of the most important choices the optimizer makes is how to retrieve data from the database. The access paths are :
- Full table scans
- table access by ROWID ,the row id of a row specifies the data file and data block containing the row and the location of the row in that block
- Cluster Scans - (a cluster scan retrieves rows that have the same cluster key value)
- Hash Scans -Oracle can use a hash scan to locate rows in a hash cluster based on a hash value.
- Index Scans-- (An index scan can be one of the following types: Unique scan, Range scan, Full scan, Fast full scan(INDEX_FFS), Index join (OPTIMIZER_FEATURES_ENABLE or INDEX_JOIN), Bitmap

How the CBO Chooses an Access Path

- The available access paths for the statement.
- The estimated cost of executing the statement using each access path or combination of paths.

Rule Based Optimizer

- The optimizer chooses an execution plan based on the access paths available and the ranks of these access paths.
- Path 1: Single Row by Row id
- Path 2: Single Row by Cluster Join
- Path 3: Single Row by Hash Cluster Key with Unique or Primary Key
- Path 3: Single Row by Hash Cluster Key with Unique or Primary Key

- Path 4: Single Row by Unique or Primary Key
- Path 5: Clustered Join
- Path 6: Hash Cluster Key
- Path 7: Indexed Cluster Key
- Path 8: Composite Index
- Path 9: Single-Column Indexes
- Path 10: Bounded Range Search on Indexed Columns
- Path 11: Unbounded Range Search on Indexed Columns
- Path 12: Sort-Merge Join
- Path 13: MAX or MIN of Indexed Column
- Path 14: ORDER BY on Indexed Column
- Path 15: Full Table Scan

Sort Operations and Joins

Sort-Merge Join

- Oracle can only perform a sort-merge join for an equi-join.
- Oracle sorts each row source to be joined if they have not been sorted already by a previous operation. The rows are sorted on the values of the columns used in the join condition.

Hash Join

- Oracle can only perform a hash join for an equi-join. Hash join is not available with the RBO.
- Oracle performs a full table scan on each of the tables and splits each into as many partitions as possible based on the available memory

Cluster Join

- Oracle can perform a cluster join only for an equi-join that equates the cluster key columns of two tables in the same cluster.

Join Method Selection

- A nested loops join (NL) is inefficient when a join returns a large number of rows [typically, more than 10,000 rows is considered large], and the optimizer may choose not to use it.
- The cost of a nested loops join = access cost of A + (access cost of B * number of rows from A)
- If you are using the RBO, then a merge join is the most efficient join when a join returns a large number or rows.
- The cost of a merge join = access cost of A + access cost of B + (sort cost of A + sort cost of B)

- If you are using the CBO, then a hash join is the most efficient join when a join returns a large number or rows.
- Estimated costs to perform a hash join = (access cost of A * number of hash partitions of B) + access cost of B

Advantages

- An execution plan shows how DBMS is going to process a statement.
- This helps tune statements = Better Performance
- Builds SQL tuning skills = $$$
- Builds Application tuning skills = $$$
- When used by developers early in the development cycle, can uncover data model, index, and performance issues before its too late!

Evaluation

Select attr1, atr2 from employee, accounts where attr3<30 and atr3>10

Output: pr attr1, atr2 (se attr3<30^atr3>10 (employee, accounts))

INPUT RELATIONAL ALGEBRA EXPRESSION:
pr attr1,atr2 (se attr3<30^atr3>10 (employee , accounts))

EXECUTION PLAN

1 EXTERNAL SORT MERGE ON attr1
 2 BLOCK NESTED LOOP JOIN
 3 LINEAR SEARCH WITHOUT INDEXES ON attr3
 4 B+ TREE SECONDARY INDEX ON atr3

STATISTICS

NOS OF I/O READS: 20270
NOS OF SORTS: 1

References

1. Database System Concepts, 4th Edition by Korth, Sudarshan, Silberschatz.

2. Presentation generated by Mahindra British Telecom DBA based on Oracle’s Query Optimizer.