Guidelines For Better Queries Performance In Database Systems

Dr. Lamyaa A.S. Al-Sammaraie
Computer Science Department
Al_Mustansiriya University

April /2006

Guidelines For Better Queries Performance In Database Systems

1. *Abstract*

Query optimization is more than transformation and query equivalence. Database designers must develop a robust cost metric, and consider cost and search space. Building extensible enumeration architecture is important, understanding the existing framework makes effective query optimizations.
This paper presents effective guidelines to Database designers to guide them for a better understanding to effective query optimization factors . It identifies and presents , effective guidelines for Query evaluation to the mostly used databases (Relational, Deductive, Object Oriented, Multiple , Distributed & Parallel) types approaches that affect optimizing relational query minimizes the most relevant performance measures (like CPU, Memory, resource usage,… etc).

2. *Introduction*

The most useful & effective facility that any DBMS provides is the Query Language . Since any data retrieved from a database requires this facility this paper is relevant to all users who access databases from end-users to the developers to the DBA.
When efficient Query approaches & accessing techniques are used , the results are highly scalable, flexible & manageable. When inefficient approaches are used response time are longer, programs long runner & application outages can occur.

3. *History & Motivation*

- Databases were widely used by the early 1980's
- Difficulty in using queries effectively
- Search for consistent queries led to first experiments with query
- optimizing data volume make performance to be considered more important. So, it is necessary to survey efficient approaches for query execution engines to execute queries in an optimal fashion.

4. *Query Processing Phases*. [3]

There are four separate phases :-

- **Parsing**
  - Query is parsed into an internal form
- **Optimization**
  - query is validated against the metadata to ensure that it only contains valid reference
  - query optimizer maps query expression into an optimized execution plan
  - using tree traversal algorithm to translate the plan into a representation for execution engine
- **Code Generation**
  - Compiled, or semi compiled, or interpreted into language.
- **Execution**

5. *Query Optimization*

Query optimization is the process of selecting the most efficient plan from many possible strategies.

➢ *Issues Of Query Optimization*

- What plans to consider
- How the cost of a plan is estimated

➢ *Goal:*

Is to find the best possible plan

➢ *Techniques*

Algorithm for evaluating relational operator:

- Indexing
- Iteration
- Partitioning

➢ *System Catalogs*

Provides information about the indexes and relations of a given query.

Catalog mostly consists of:

- # distinct key value for each index
- # of tuples for each relation

Catalogs must be updated periodically

➢ *Qualities of a good Query Optimizer*

- Search space must be able to include plans that have a low cost.
- The costing technique must be accurate.

The enumeration algorithm that searches through the execution space must be efficient.

6. *Investigated Data Bases*

Query Optimization in the following Databases will be investigated in this research:-

- Relational DBs
- Deductive DBs
- Object Oriented DBs
- Multiple and distributed DBs

Although **the relational model** is still the most widely-used model for databases, other types of databases exist for both research and business purposes.

➢ *Relational Database*

- Cost estimation
  - Estimated time to execute possible queries
  - Based on I/O access and projected CPU time
  - Maintained current list of table dependencies
- Produced left-deep plans
  - Only binary joints are considered
  - For a join of k relations, optimizer only considered plans that join kth relation with a subset of k-1
  - Result: left-to-right-associative statements in relational algebra, where only one relationship at a time is considered
- How Does the Optimizer Select A Table?
  - Selectivity: Ratio of key values to items in table
  - Clustering: Order of items in physical storage
  - Taken together, these indicate how many page fetches and how much branching the query needs

- Joining Tables , there are two options:
  - Scan first table for values, join with second based on the second's index
  - Sort rows of both tables by join fields, then search for matching values

Depends on whether or not the tables have indices [16]

➢ *Deductive Databases*

- Similar to relational databases
- Query language has additional capability
- Can express recursive queries
- Use logic programming to support complex reasoning
- Integrate rules and facts of traditional DBS
- Used in supporting advanced applications in the areas of:
  - Computer aided design
  - Manufacturing
  - Office systems

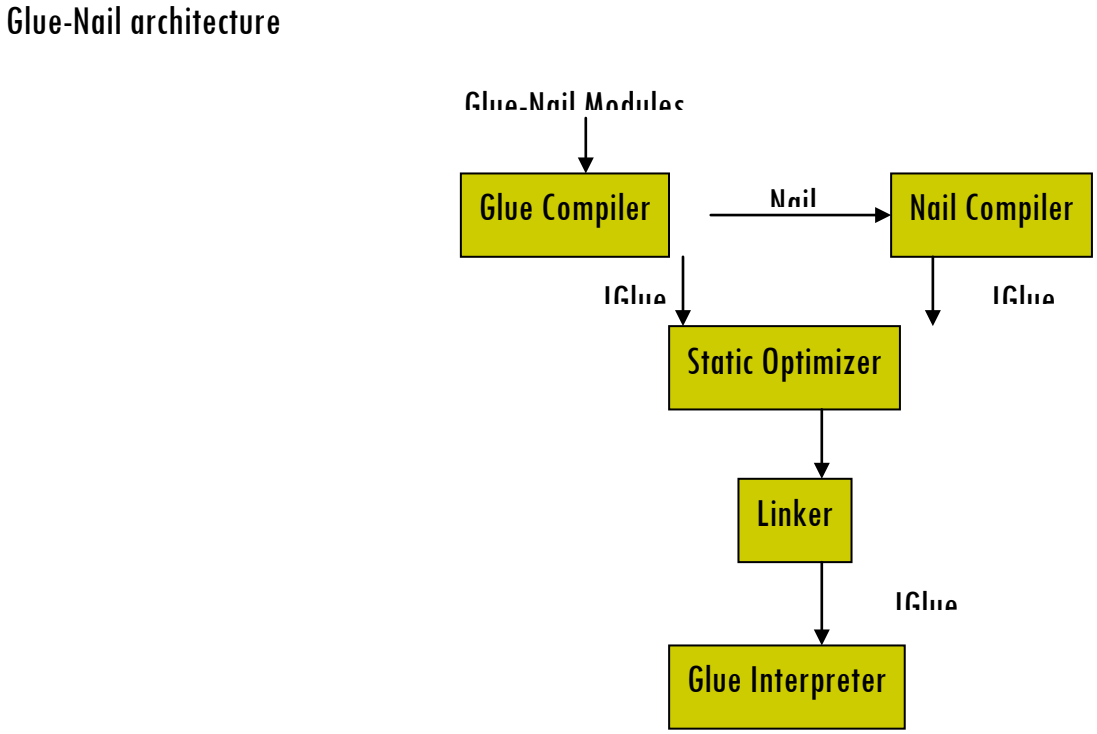
A language called GLUE developed for logical rules, has the power of SQL statements as well as a conventional language for the construction of loops ,procedures & modules, next is a brief of this language :-

Glue-Nail NAIL (Not Another Implementation of Logic) will be investigated in this paper

- Ex: Glue-Nail DB have two complementary languages:
  - Declarative and procedural
  - Glue: Procedural language having query capabilities of
    - Updating, aggregation, input-output libraries, and procedural control
  - Nail: logic-based query language and uses
    - Function symbols to represent complex objects
    - Express powerful recursive queries
- Glue is a lower level language compiled in Glue-Nail

- The query optimization occurs in evaluation of the Glue code[9]

The next diagram illustrates the architecture og the GLUE-NAIL:-
Glue-Nail architecture



Deductive Databases, **Glue-Nail**

- Optimizers used in Glue-Nail system:
  - Runtime query optimizer:
    - Improves system performance by reoptimizing query evaluation plan automatically
    - Creates and drops indexes automatically
  - Glue adaptive optimizer:
    - Effective alternative reoptimization
    - Has a large performance gain [9]

➢ *Object-Oriented Databases (OODB)*

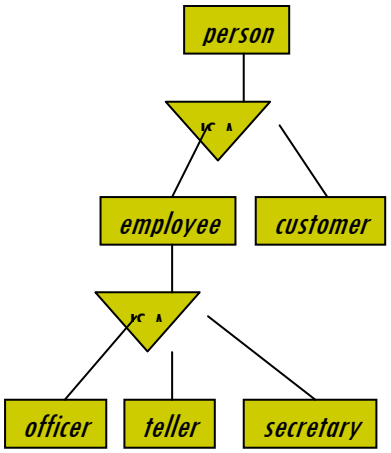
Object oriented databases emerged in the mid-80's in response to the feeling that relational databases were inadequate for certain classes of applications.
The advantages of the object database approach are that applications which define a rich type system for their data structures can carry this over to the database when these data structures are made persistent.
The impedance mismatch is eliminated as the database becomes a "persistence extension" to the language rather than an external service that one has to talk to through a narrow and limited interface.
Weaknesses

- procedural navigation
- querying breaks encapsulation (or is limited)
- what about closure?
- no mathematical foundation

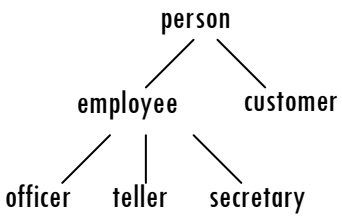
Object systems introduce pointers linking objects to each other. This in turn promotes a procedural style of navigation among the data items that can be seen as a step backwards from the higher-level declarative approach introduced by relational systems.
While encapsulation (the idea that an object's internal state is only accessible through its designated methods) is a valuable modular technique, it presents problems with queries.
appropriate method has been written. It could be summarized by:-

- OODB combines the data abstraction and computation models of object-oriented programming languages C++/Java
- Contain the performance and consistency features of databases
- Developed to support powerful application domains
    - Engineering databases
    - Office information systems
- Currently much attention is given experimentally and theoretically

The following is an example of E/R diagram that is used to represents (OODB)



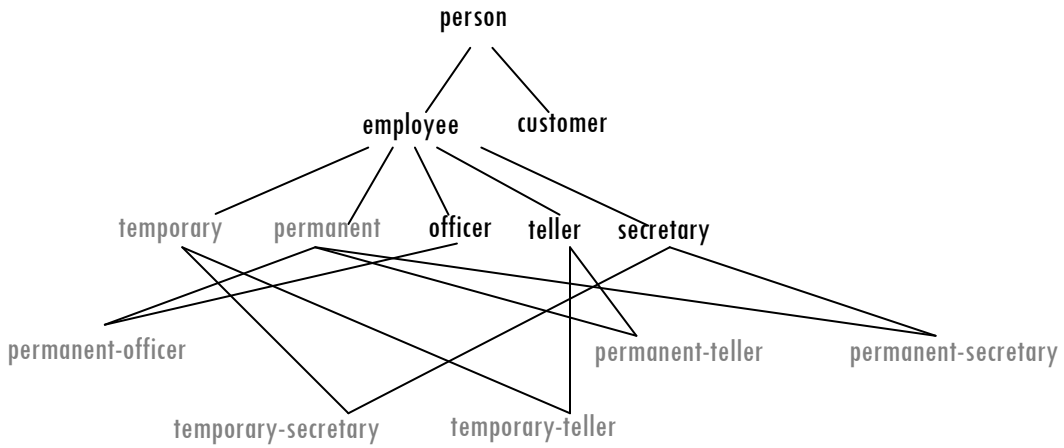
Specialization hierarchy



Class hierarchy

```
class person{
string name; string address; string get-name();
string getAddress();
int set-address(string new-address);
};
class customer isa person{
int creditRating;
};
class employee isa person { ……. };
class officer isa employee { …… };
class teller isa employee { …… };
class secretary isa employee { …… };
```

Definition of class hierarchy in pseudo code



Class hierarchy

Multiple inheritance usually happen

- Implementation of query optimization is a delicate process
- Need to find an execution plan that minimizes the cost
- Logical and physical levels are traditional in optimization
  - Logical: semantic properties (query writing)
  - Physical: find the best algorithm using the cost model [10]

➢ *Parallel Databases*

- Parallelism demonstrates the following two properties :-
  - o linear *speedup* where adding n times as many processors reduces response time by a factor of n.
  - o linear *scaleup* where adding n times as many processors allows the system to perform a task n times the size in the same amount of time.
- Three problems exist which hinder parallelism in obtaining linear speedup and scaleup:
  - o *Startup*: Starting a parallel system may require the creation and coordination of thousands of processes. If this task dominates the actual query processing time then an inefficient system results.
  - o *Contention*: As more processes are created, contention for shared resources increases. This slows down the system.
  - o *Skew*: Skewed data can lead to unbalanced load so that only a few processors are working while the rest are idle.
- A perfect algorithm on a parallel machine would scale easily at a low-cost, and would demonstrate linear speedup and scale up even for hundreds or even thousands processors. Three architectures have emerged from the quest for perfect parallel machine; they are :
  - o shared-memory (SM),
  - o shared-disk (SD)
  - o shared-nothing (SN)

In the shared-memory architecture, each processor has access to all disks in the system and to a global shared memory. The main advantage of a shared-memory system is that it has zero communication cost; processors exchange messages and data through the shared memory. This also makes it easier to synchronize processes. Another advantage of such a system is that load-imbalance can be corrected with minimal overhead. The drawback of such a system is the problem of contention. A

shared-memory architecture can only be scaled to few processors (30-40) before interference begins to affect the rate at which the memory can be accessed.
Shared-nothing is the extreme opposite; every processor has its own memory and disks. Nodes communicate by passing messages to one another through an interconnection network. The main advantage of shared-nothing is its ability to scale to hundreds and potentially thousands of processors. The drawbacks of such a system are complications created by load-imbalance and its dependence on a high bandwidth interconnection network for effective message exchange. Parallelizing a sort or join operation may gain little in response time if the data is highly skewed, and the operation depends a great deal of coordination among the nodes. Another disadvantage to shared-nothing architecture is the problem of node failure. If a processor fails, then all access to the data owned by that processor is lost. A solution to the node-failure problem is a shared-disk architecture where every processor can read and write to any of the disks in the system but manages its own memory. Shared-nothing has emerged as the design of choice, mainly due to its high reliability and ease of scalability to hundreds of processors. Of course one of the main issues is cost, and since shared-nothing architectures can be constructed from existing sequential machines using a simple interconnection network, it is the most cost effective means of parallelizing a database system. Most of the current research in the field of parallel databases concentrates on shared nothing architectures; In examining the way in which queries are executed on parallel systems we observe two forms of parallelism:
Intra-operator parallelism: one operation is parallelized over several processors. On a shared-nothing system this is achieved by partitioning or de-clustering the data among the processors. When an operation such as a scan is performed each processor operates on its local data cluster, the results of each processor are then combined for the final result (which can also be partitioned among processors). There exist several methods for de-clustering data; these include round-robin distribution, range partitioning and hash partitioning.

Inter-operator parallelism: several operations are executed concurrently. Each of these operations is assigned to one or more processor. When more than one processor is used for a given operation, then we are also exploiting intra-operator parallelism. There are two forms of inter-operator parallelism: independent and pipelined. Independent means that no dependencies exist between operations, pipelined refers to a producer – consumer relationship between operators. The result of the producer operator is pipelined to the input of the consumer operator instead of being materialized. This has the obvious advantage of not having to wait for the input data to be completed before starting a new operation on the data.
Notice that in the above discussion we refer to the parallelism of operations while not clearly defining the specifics of an operation. This is because there exists varying degrees of granularity for parallelism, depending on how small or how big are the pieces used to construct a query. On one end of the scale we can parallelize entire queries, referred to as course granularity, and on the other end we parallelize each of the components, for example a hash-join: build, scan and probe, this is called fine granularity which plays an important rile in query optimization.[10]

➢ *Multiple and Distributed Databases*

- Multi database system (MDBS): An additional software layer on a heterogeneous database system that allows access and manipulation of information.
- Distributed database systems (DDBS): A collection of logically-interrelated databases distributed over a computer network.
- Multi database system (MDBS): A distributed system that acts as a front end to multiple local database management systems (DBMS).

- Several levels of (request and data) transactions are needed,
- different local capabilities are assumed,
- more dynamic query optimization techniques are needed,

- local query optimization information may not be known at the global level,
- response time for local request may not be predictable,
- more constraints about where data can be transferred to and where an operation can be performed, need to be considered during global query optimization.
- Many distributed query optimization tools are not suitable for MDBS.
- Two-phase optimization approach, along with several global query optimization techniques, are used in MDBS.
- Global query optimizations:
  - semantic, parametic, and adaptive query optimization. [16]

7. *Other Query Optimization Issues*

- Memory is an issue when determining execution plans.
- Databases involve multimedia, and web uses Optimizers in fuzzy queries.
  - Fuzzy queries are flexible, easy-to-use design
  - Have information beyond the algebra and relational operators.
- SQL extensions for decision support systems are widely used.
- Query optimizers are used in decision support systems
  - uses cube as an extended language.
  - Cube/data cube: operator that generalizes the histograms, cross-tabulation, roll-up, drill-down, and sub-total constructs found in most report writers.[6]

8. *Conclusion*

- Query optimization more than transformation and query equivalence.
- Database designers must develop a robust cost metric, and consider cost and search space.
- Building extensible enumeration architecture is important
- Understanding the existing engineering framework make effective query optimizations.
- The guidelines in this paper will help database designers via guiding them for a better understanding to effective query optimization factors .

9. *References*

1. ACM Vol. 27,No.1,“Transmissions on Database systems”, march/2002
2. Aditi Group, Dept. of Computer Science & Software Engineering, The University of Melbourne, “ The Aditi deductive database “ Jun/2005
3. Adrian-Constantin Oneţ **“**Natural language query for deductive databases **“** Ph.D. thesis 2003
4. Dates,C.J.”An Introduction to Database Systems” 2002
5. Ellis Cohen,”The Theory, Practice & Methodology of relational Database Design and Programming” , 2005
6. FRANK, MANOLA, “An Evaluation of Object-Oriented DBMS Developments, 2003 Edition”
7. Jiawei Han, “Constraint-Based Query Evaluation in Deductive Databases” 2002
8. Fredr,Mcfadden “Modern Database Management” sixth ed. 2000

9. Jiawei Han “Query evaluation techniques in deductive atabases”,2003
10. Kim ,W. And Lochovsky (Eds), “Object Oriented Concepts, Databases and Applications”,2004
11. Max Kremer & Jarek “A survey of query optimization in parallel databases”, Technical report 1999-2004
12. Mengichi liu “ A Logical Foundation in deductive Object oriented database” 2002
13. Mnushkin, Dima “Object oriented technologies” 2004
14. [sherry I M. Larsen] ”Top Ten SQL Performance Tip” , June , 2005
15. Stagano, Frank “A Gentle introduction to relational and object oriented Databases”, 2002
16. Stefano Ceri Palagatti “ Distributed Database Principles & Systems 2000
17. W. KIM AND LOCHOVSKY (EDS), “Object-Oriented Concepts, Databases, and Applications”,2000

خطوط ارشادية لاداء افضل للاستعلام في نظم قواعد البيانات

د. لمياء عبد الصمد السامرائي

المستخلص

يستنتج هذا البحث أهم العوامل المؤثرة في استجواب قواعد البيانات ويبحث في هذه العوامل ويقدم خطوط ارشادية لكل نوع الغرض منها توفير افضل اسلوب يزيد من كفاءة الاداء عند اجراء عمليات الاستعلام لأنواع قواعد البيانات المستخدمة حاليا وهي العلائقية والاستنتاجية والشيئية والموزعة والمتوازية ان تأثير هذه العوامل متغير

كما ان فهم اطار العمل يؤدي الى استخدام طرق استعلام كفوءة مع الاخذ بنظر الاعتبار في مرحلة التصميم ان العديد من عوامل الاداء كزمن الاستجابة ، وقت تنفيذ البرامج وتوقف التطبيق الفجائي تعتمد علىعوامل عدة أهمها كفاءة عملية الاستعلام .