Challenges of Modern Query Processing

Archana Kumari and Vikram Singh

Abstract Efficient query processing plays a critical role in numerous settings especially in case of data centric applications. Starting from the architecture to the final stage of result compilation a query processing system has to undertake several challenges. In this paper, we compare different architectures and existing approaches of query processing. In modern days, database systems have to deal with data distribution. To make challenges more complex, the participating databases might be heterogeneous in nature. In this paper, we discuss various challenges of query processing systems in centralized, distributed, and multidatabase backgrounds. We also analyze various parameters and metrics that directly impact query processing.

Keywords Distributed database systems · Multidatabase systems · Query optimization · Query processing architecture · Query processing challenges

1 Introduction

One of the key motivations behind the use of database systems is the aim for integration of functioning data of an enterprise and to deliver unified, thus controlled access on the data. Data is retrieved by posing a query to the database system. A query is a language construct that helps to retrieve a part of data from database [1]. When a user frames a query, a query processor frees the user from specifying the exact procedure to get the required answer via various transparency mechanisms. It takes a query, parses it, optimizes it, and finally retrieves the intended result. Query processing performs transformation on a high-level query and converts it into low-level query. This transformation must be complete and

A. Kumari (✉) · V. Singh
Computer Engineering Department, National Institute of Technology,
Kurukshetra 136119, Haryana, India
e-mail: archana19dhankar@gmail.com

V. Singh
e-mail: viks@nitkkr.ac.in

correct [2]. The Main aim of query processing is to find transformations that are most efficient for generating query result. In centralized databases, query processor finds the most efficient equivalent relational algebra query. While in distributed systems, the query processor takes into account the cost of data transfer.

The main contribution of the paper is to discuss various aspects of query processing and related modern-day challenges in it. The paper has illustrated existing architectures of query processing in Sect. 1 and subsequently highlight query processing in the centralized database systems and distributed systems in Sects. 2 and 3, respectively. The inherent challenges in modern-day query processing are discussed in conclusion, in Sect. 6.

1.1 Query Processing Architecture

Architecture of a query processing system determines the internal representation of query, enumeration of query execution plans, and the phase at which system has the

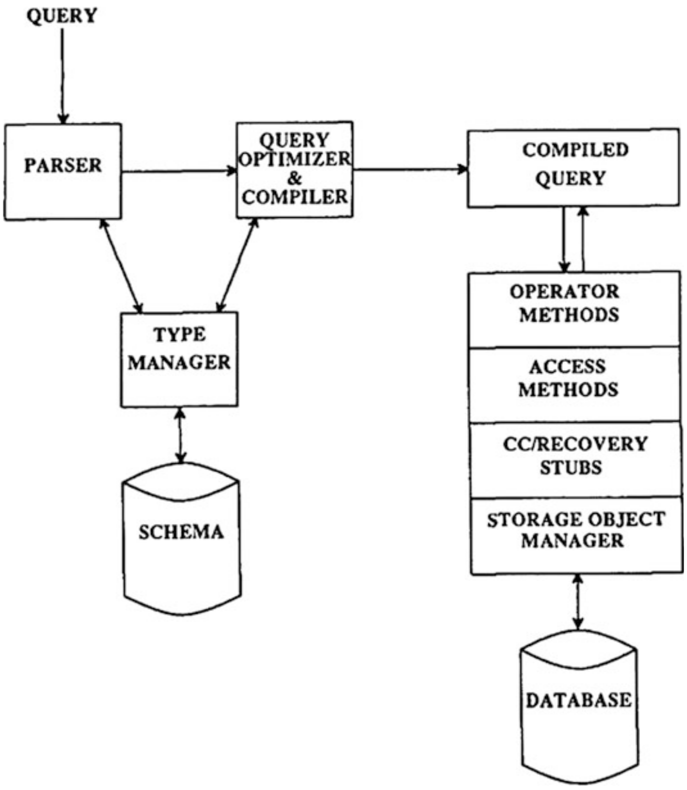

Fig. 1 Query processing phases in Carey et al. architecture

Table 1 Comparison of Haas architecture and Carey architecture for query processing

Parameter	Haas et al. architecture	Carey et al. architecture
Extending DBMS	Customizable System—The relations of a query language can be modified and can be made more comprehensive using new operators. For instance, In STARBURST [19], the query rewriting part can be modified by adding new rewrite rules	Toolkit Systems—They have libraries and modules that offer various alternate methods to perform a task. For Example, In EXODUS [20], architecture type independent access methods, like B-tree, linear hashing, can be extended
Phases	Query rewriting and query optimization are two different phases	Query rewriting and query optimization are merged in one phase
Cost availability	The query rewrite phase does not have the cost information available	All transformations are algebraic and cost-based
Internal Representation	The building block in this case is an enhanced query graph	Query trees and operator trees are used for all phases of optimization

knowledge of the cost of transformations. These factors are crucial in evaluating the complexity of the system. There had been many proposed architectures, two of them are Haas et al. [3] and Carey et al. architecture [4].

Haas Architecture: The main component is parser that translates the input query into an internal representation like query graph. Query rewrite transforms a query to carry out optimizations. Query Optimizer enumerates several plans that specifies how given queries should be executed. Plan Refinement and Code Generation converts the plan made by query optimizer into executable code. Catalog stores the necessary information for parsing, rewriting, and optimizing a query. Query Execution Engine gives implementation details for the operators used.

Carey Architecture: In addition to the Haas Architecture, the main component is storage object manager for providing the capabilities for modifying storage objects. Access methods give associative access to all files of objects and support for versioning. Schema-related information in the databases is provided by type manager to support a wide variety of applications. The various methods or operations on objects are supported by Operator Methods. These components provide the dedicated functions related to query processing. The schematic architecture of Carey for query processing is shown in Fig. 1 (Table 1).

2 Centralized Database Systems

Query processing has three steps: decomposition, optimization, and execution. Query decomposition is done by query compiler. Query compiler goes through several phases like parsing, checking for semantics, query rewrite [5]. Compiler

Table 2 Comparison of bottom-up and top-down query optimization process

Parameter	Bottom up	Top down
Query representation	Generally a query graph is used for internal representation of the queries	Relational calculus expressions are normally used for internal representation
Find upper and lower bounds	The upper/lower bounds are not available to bottom-up optimizers, since such optimizers generate the plans before they considered the larger containing plans	Top-down optimizers are superior allowing to this degree because they can use upper and lower bounds to avoid generating groups of plans
First step in optimization	First, an internal query representation is made so that user queries can easily be mapped	In the first step, the algorithm constructs an access plan for all tables involved in the query
Advantage	The inferior plans are discarded very early	Every stage has a larger and richer set of plans to choose from
Disadvantage	This algorithm has exponential running time and space complexity, therefore, it is not viable for complex queries	Top-down optimizers normally retain all multi expressions. Due to this there is memory wastage
Example	IBM's System R project	INGRES relational DBMS

validates the syntax of the input query and returns appropriate error, if it finds any discrepancies. In query graph model of compilers, an internal representation of the query is formed. A compiler also checks for any inconsistencies in the sub parts of the input query. A compiler utilizes the global semantics to transform the query into a format that is easier to optimize [6]. Recent advancements in programming languages have led to the development of highly efficient query compilers [7].

Query optimization is the most critical phase in query processing and in centralized database systems; it is a two-step process. First is the logical optimization that involves the classic transformation rules on the algebraic trees which helps to reduce the manipulated data volume. Second is the physical optimization that includes the process of determining an appropriate join method as well as their ordering. The decisions are made by taking into account the size of the relations, the physical organization of the data, and access paths. Another approach is based on response time of a query. Response time can be defined as the time elapsed between the submissions of a user query till the time first result is generated [2]. The response time is calculated as the sum of three separate costs, first communication cost, second input/output cost, and final CPU cost.

There are generally two approaches for query optimization. In top-down approach, a query expressed in relational calculus is broken down into smaller sub queries. While, in case of bottom-up approach, many plans are built from simpler sub-plans (Table 2).

3 Distributed Database Systems

A distributed DBMS is distributed over a computer network such that its existence is transparent to the user. The participating databases must be logically related. The only means of communication is with the help of message passing [3]. In case of distributed databases, the communication cost plays the most critical role.

3.1 Challenges in Distributed Database Query Processing

This data distribution makes the query processing a multifaceted procedure [2, 3, 8, 9].

Fragmentation and Replication: The distributed DBMS has to take into account all the fragments and the replicated copies relevant to the query.

Cost Model: The cost model selection is application specific. A database system can choose to follow the path of minimum resource consumption or minimizing the response time.

Search space Exploration and Exploitation: The search space consists of all the equivalent query execution plans that are made after using various transformation rules. The search space could grow exponentially larger once the number of relation involved in the query increases. The search strategy has to be really efficient since there could be a very large search space available.

Control Site Selection: The appropriate control site for final query processing should be selected carefully.

Where and When to Optimize: The timing and placing of the optimization process is deciding factor of a distributed DBMS performance. The alternate execution plans can be made at compile time and the optimizer can select from the available options at run time. The phases like parsing and rewriting of the query can be done at client side while QEP (Query Execution Plans) refinements can be done at server site. In crowd sourcing systems the complexity is relatively high [10]. Hence, the system must free the user from over burden and is solely responsible for compilation, generation of execution plans, and evaluation of the marketplace.

3.2 Challenges in Distributed Database Query Execution Process

Once all the execution plans are generated, the most optimal plans are selected and the rest are pruned at successive stages. The real challenge now is to execute these plans efficiently [2, 3, 8, 9].

Selecting the data transfer mode: Typically row blocking technique is used. In this, a whole block of data is transmitted rather than a tuples.

Use of cache results: Whether to use the data caches that temporarily store copies of data to utilize the benefits of temporal locality.

Organization of sites on the network: The sites are typically arranged in a hierarchical way so sending and receiving costs can vary significantly [11–13].

Optimization of Multicasts: Multicast communication can be done to increase the degree of parallelism.

Use of semi joins: Semi joins reduce the size of intermediate results, since it is based on exit clause. However, this method is suitable where join selectivity factor is good.

Execution Control: The control site could be one single site that is centralized control or a group of sites can act as control sites [13].

The final challenge in query processing is to combine the various building blocks and build a full-fledged efficient query execution plan [2].

4 Multidatabase System

Multidatabase systems (MDBS) are a collection of individual DBMSs that are fully autonomous and cooperate at their own will [2, 3]. In case of multi database, the global conceptual schema is only a sub-part of the logical integrated schema formed by the individual schemas of component databases.

4.1 Challenges in Multidatabase Query Processing

Query processing in a multidatabase system is highly complex and requires more efforts than in a distributed DBMS [2, 3].

Computing Capability: The participating DBMSs may have varying computing capabilities. This hinders the uniform distribution of the query distribution process.

Cost Function: The participating DBMSs may have different cost functions. Hence, the local optimization is different for the same query posed.

Language Models: The participating DBMSs may have different language models for example relational, object oriented, and XML. This creates a problem in translating the query to component DBMS and in integrating heterogeneous results.

Autonomy: The participating DBMS exhibit autonomy. The autonomy can be defined across three dimensions: Communication, design, and execution autonomy.

Semi–Join-based optimizations: The semi-join-based optimizations are extremely difficult, since the source and the target tables are present on different component DBMS. The retrieval of the join attributes and shipping them from target site to host site, performing the join operation at the source site and modifying the target relation DBMS adds complexity.

Architectural Difficulties: The architecture of the multidatabase system is quite complex. The design of wrappers is difficult, since it has to take into account the translation art of the queries. With any change in local schemas, the wrapper and mediator have to be modified accordingly.

Data distribution: The query needs to reach every possible source to obtain the maximally contained query. For systems that involve continuous processing of data streams query optimization becomes much more complex, since the data never ceases to come to the databases. In such cases to efficiently overcome the heterogeneity event processing middlewares [14] are used.

Integration of results: After obtaining the results for an input query, it is sent back to the mediator to carry out necessary translations. The integration of the heterogeneous results thus obtained is a daunting task.

5 Query Processing: Work Done So Far

Query processing has become a major area of research and study in various computing applications. Earlier query processing was confined to single databases stored at a single geographical location. The increase in hardware and software systems has given modern query processing immense power over traditional query processing systems [15]. These days, databases must adapt to this variety of options available. Modern query processors have to make decisions and must deliver advance utilities and services. Due to availability of huge amount of data, modern query processors must provide scalability [16]. Traditional query processor did not exploit parallelism. They neither have the expertise nor potential to perform operations any faster. Modern query processor must provide quick and effective results to meet the never ending demands of the user. Modern query processors are capable of processing time-continuous data [17, 18]. Traditional query processor used to select a single query execution plan out many possible, while modern query processors have the capability to change plans to deal with non-uniform distributions of data. Modern query processing has certainly come a long way in efficiency, reliability, and performance. Table 3 summarizes the available surveys on query processing.

6 Analysis and Challenges

The recent ground breaking developments in database systems led to spectacular results in data management. Organizations, these days find it difficult to find approaches to manage their databases efficiently, conveniently, and in a more controlled fashion. The performance and efficiency of a database system can be analyzed across various metrics.

Table 3 Surveys on query processing

Title and authors	Aim
Donald Kossmann: The State of the art in distributed query processing [3] (ACM Computing Survey, 2000)	This paper presents a textbook architecture for distributed databases. It also discusses challenges faced in query optimization and query evaluation. A brief introduction is also present about multi database
Yu and Chang: Distributed Query Processing [21] (Computing Surv., Vol. 16, 1984)	Various heuristic based, semi-join-based both enumerative and non-enumerative query optimization techniques are used. Types of query, like cyclic and tree queries, are also discussed
Matthias Jarke and Jurgen Koch: Query Optimization in Database Systems [1] (Computing Surv., Vol. 16, 1984)	A variety of methods that are logic and heuristic based are discussed. The focus of the paper is on query optimization in centralized databases
Liu and Yu: Performance Issues in Distributed Query Processing [9] (IEEE transactions on parallel and distributed systems, 1993)	Various performance issues of query processing are discussed. It is shown that these algorithms can be divided into algorithm-based issues and implantation-based issues
Tamer Özsu and Patrick Valduriez: Principles of Distributed Database Systems [2] (Computing handbook Springer ISBN-978-1-4419-8833-1, Springer, 2014)	This handbook is about everything related to distributed database from database design to query processing in detail. Every aspect of the distributed database both in homogeneous and heterogeneous environment is presented in minute details

6.1 Analysis of Impact Parameters

Response Time: While analyzing the performance parameters in case of distributed system all three cost namely communication, I/O, and CPU time plays equal role.

Cost: Since various methodologies like parallel and distributed processing has become a possibility. The communication cost is much larger than I/O and CPU cost in case of distributed databases.

Heterogeneous databases: The query processing optimization is done into two phases. One is the global optimization that is understood by component databases and the second phase is localized optimization implemented individually.

Modern-day database software: The database software available these days are much more stable and highly user friendly.

Advanced Hardware technology: Also, in this modern era, which we can rightly call as digital age, the cost of hardware has gone down and the efficiency has gone up

Network: Due to availability of fast network the query processing approaches have to give equal weightage to all the three costs—communication, I/O, and CPU costs.

Bandwidth: Availability of larger bandwidth provides faster and more reliable connections between the distributed sites.

6.2 Inherent Challenges of Data

Talking about the challenges faced by databases, we have to admit that we are now living in the “ocean” of data. We have to consider various aspects of data analysis.

Data Dimensions: The size of data is too large to be efficiently handled by conventional databases. Hence, the need of the hour is to enhance the features of the present database systems which can incorporate as much data as possible.

Data Stride: The data is huge not only in dimensions but the pace is high too. Every day, millions and millions of data feeds are coming to the system, making it impossible for the systems to adapt at the same rate.

Data Diversity: One query processing approach might be considered for a given type of data set while the other for other dataset.

Data Changeability: In this modern age, we need database systems that can handle variability of data efficiently. The inherent interpretations of this massive amount of raw data depend on the underlying context. For example, in case of natural language processing, a single word may have different meaning.

Data Accuracy: The user will get inaccurate results, if the principal database gathers the data from diverse sources which might not be complete. In such cases, query containment concept is used. The user will get a subset of the complete answer.

Data Envisioning: One of the core tasks of a database system is to present the actual result of the query under processing in a readable and understandable format.

All of these factors demand careful examination, in particular for enterprises not already on the efficient query processing systems bandwagon.

References

1. Jarke, M., Koch, J.: Query optimization in database systems. ACM Comput. Surv, vol. 16 issue 2, pp. 111–152 (1984)
2. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems: Third Edition Springer Science (2011)
3. Donald Kossmann: The State of the art in distributed query processing. ACM Comput. Surv., vol. 32, issue 4, Dec. pp. 422–469 (2000)
4. Carey, M.J, DeWitt, D.J., Frank, D., Graefe, G., Richardson, J.E.: The Architecture of the EXODUS Extensible DBMS: A Preliminary Report, International Workshop on Object-oriented Database Systems, pp. 52–65 (1986)
5. Aho, A.V., Lam, M.S., Sethi, R., and Ullman, J.D.: Compilers: Principles, Techniques and Tools, Pearson Education, Second Edition (2006)
6. IBM Knowledge Center, https://www.ibm.com/support/knowledgecenter

7. Rompf, T., Amin, N.: Functional pearl: a SQL to C compiler in 500 lines of code, ACM SIGPLAN, vol. 50 Issue 9, pp. 2–9 (2015)
8. Ceri, Pelagatti: Distributed Databases: Principles and Systems, McGraw-Hill (1984)
9. Liu, C., Yu, C.: Performance Issues in Distributed Query Processing, IEEE transactions on parallel and distributed systems, vol. 4, issue 8, pp. 889–905 (1993)
10. Fan, J., Zhang, M., Kok, S., Lu, M., Ooi, B.C.: CrowdOp: Query Optimization for Declarative Crowdsourcing Systems, IEEE Transactions on Knowledge and Data Engineering, vol. 27, issue 8, pp. 2078–2092 (2015)
11. Kumar, T.V., Singh, V., Verma, A.K.: Generating distributed query processing plans using genetic algorithm. In: Proceedings of the 2010 International Conference on Data Storage and Data Engineering, pp. 173–177. IEEE, USA (2010)
12. Singh, V., Mishra, V.: Distributed query plan generation using aggregation based multi-objective genetic algorithm. In: Proceedings of International Conference on TCS, pp. 20–29. ACM, USA (2014)
13. Mishra, V., Singh, V.: Generating optimal query plans for distributed query processing using teacher-learner based optimization. In: Proceedings of 11th International Conference on Data Mining and Warehouse, vol. 54, pp. 281–290. Elsevier, India (2015)
14. Pinnecke, M., Hobach B.: Query Optimization in Heterogenous Event Processing Federations, Datenbank-Spektrum, vol. 15, issue 3, pp. 193–202 (2015)
15. Tomas, K. T., Hille, M., Ludwig, M., Habich, D., Lehner, W., Heimel, M., Markl, V.: Demonstrating efficient query processing in heterogeneous environments, ACM SIGMOD International Conference on Management of Data, pp. 693–696 (2014)
16. Doulkeridis, C., Norvag, K.: A survey of large-scale analytical query processing in MapReduce, The VLDB Journal, Volume 23, Issue 3, pp. 355–380 (2014)
17. Nehmea, R., Worksb K., Leib, C.: Multi-route query processing and optimization, Journal of Computer and System Sciences, vol. 79, issue 3, pp. 312–329 (2014)
18. Agarwal, S., Milner, H., Kleiner, A., Talwalkar, A.: Knowing when you're wrong: building fast and reliable approximate query processing systems, ACM SIGMOD International Conference on Management of Data, pp. 481–492 (2014)
19. Pirahesh, H., Hellerstein, J., Hasan, W.: Extensible/rule based query rewrite optimization in starburst, ACM SIGMOD Conf. on Management of Data, pp. 39–48 (1992)
20. Graefe, G., Dewitt, D.: The EXODUS optimizer generator. In: ACM SIGMOD Conference on Management of Data, pp. 160–172 (1987)
21. Yu, C.T., Chang, C.C.: Distributed Query Processing, ACM Computing Surveys (CSUR) Surveys, vol. 16, issue 4, Dec., pp. 399–433 (1984)