Smart Search Methods in Expert Database Systems

Wael Said[1], M.M.Hassan [2,] Amira M. Fawzy [3]
[1]*Computer Science Department,*
[2]*Professor on Information System Department,*
[3]*Information Systems Department.*
Faculty of Computers & Information,
Zagazig University, Egypt.

Abstract

Expert database is the integration between database technology and techniques developed in the field of artificial intelligent (AI), this integration cause database became more complicated and complex. This is in addition to the tremendous technological advances that have recently led to the fact that databases have become very large and complex. From these data, to reach valuable information you need more effort and cost. Most researchers have focused on using optimizer to address this problem, but this solution is costly and not satisfactory in all cases. Therefore, this research discusses different intelligent methods that used to improves performance of the query execution and minimizes the total time that the database server spends processing requests.

Keywords - expert database system, query execution, intelligent search**.**

I. INTRODUCTION

A considerable amount of effort in the fields of Expert Systems and Database Systems has been focused on integrating the two systems. While efficient management of large amounts of knowledge is required for Expert Systems, intelligent search

techniques for query processing are demanded by Database Systems as in[1].

The motivations driving the integration of these two technologies include the need for (a) access to large amounts of shared data for knowledge processing, (b) efficient management of data as well as knowledge, and (c) intelligent processing of data.

The integration between different fields of AI, database systems and logic programming leads to the data became more complicated and complex to discover knowledge from it. More researchers concentrated on this challenge to reach valuable information from it, some of these focused on optimization of query processing and others used AI methods to detect stylized pattern exist.

This paper discusses "what is meant by expert database systems?" in section *II* and discusses the methods of search and clarifies strengths and weaknesses as in section III.

II. EXPERT DATABASE SYSTEMS

Expert database systems (EDS) are database management systems (DBMS) endowed with knowledge and expertise to support knowledge-based applications which access large shared databases. The special appeal of EDS is that they evoke a variety of ways in which knowledge and expertise can be incorporated into system architectures [2]. One can envision several possible scenarios:

- expert system loosely-coupled with a database system;
- database management system (DBMS) enhanced with reasoning capabilities to perform knowledge-directed problem solving;
- Logic programming (LP) system or an AI knowledge representation system, enhanced with database access and manipulation primitives;
- intelligent user interface for query specification, optimization and processing;
- And a tightly-coupled EDS "shell" for the specification, management and manipulation of integrated knowledge-databases.

The terms loosely-coupled means that both the AI system and the DBMS will maintain their own functionality and communicate through a well-defined interface. For example, an AI system might send SQL queries to the database system. Tightly-coupled, on the other hand, implies that at least one system has knowledge of the inner workings of its counterpart, and that special, performance enhancing access mechanisms are provided.

III. THE SEARCH METHODS USED IN EDS

A. Scalpel system

The scalpel system deal with the streams of queries problem, which this system detects the stylized pattern that exist in the streams of queries and optimizes request streams using context-based predictions of future requests. Scalpel uses its predictions to provide a form of semantic prefetching, which involves combining a predicted series of requests into a single request that can be

issued immediately. Scalpel's semantic prefetching reduces not only the latency experienced by the application but also the total cost of query evaluation [3].

Scalpel implements a kind of prefetching. We call it *semantic prefetching* because Scalpel must understand the queries Q1 and Q2 in order to generate an appropriate Qopt as in [4].

There are two reasons to do semantic prefetching.

- First, it provides the query optimizer at the server with more scope for optimization.
- Second, by replacing many small queries with fewer larger queries, Scalpel can reduce the latency and overhead associated with the interconnection network and the layers of system interface and communications software at both ends of the connection.

The scalpel systems depends upon the client send streams of requests. This system capable of discover query pattern from these requests. The system identified three types of query patterns that are amenable to optimization: batches, nesting, and data structure correlations. Scalpel associates a query context with each request in the stream. Since Scalpel's focus is on detecting and optimizing nested query patterns, the

Query context of each request is defined to be the list of queries that are open at the time of the request. Queries are listed in the context in the order in which they were opened by the application.

B. Components of the scalpel system

Fig. 1 describes the basic components of the scalpel system that shaded in the figure. These components are called monitor, pattern detector, pattern optimizer, context and rewrites and Prefetcher as in [5]. The scalpel system results are considered significant. The cost is decreased and the time an application spends waiting for database requests (exposed latency), or the total time that the database server spends processing requests is decreased. These results based on some application. And this system succeeds to achieve the prediction of the future request based on the stylized pattern.

But we need the system to achieve higher performance, less costs for processing the query and less time.

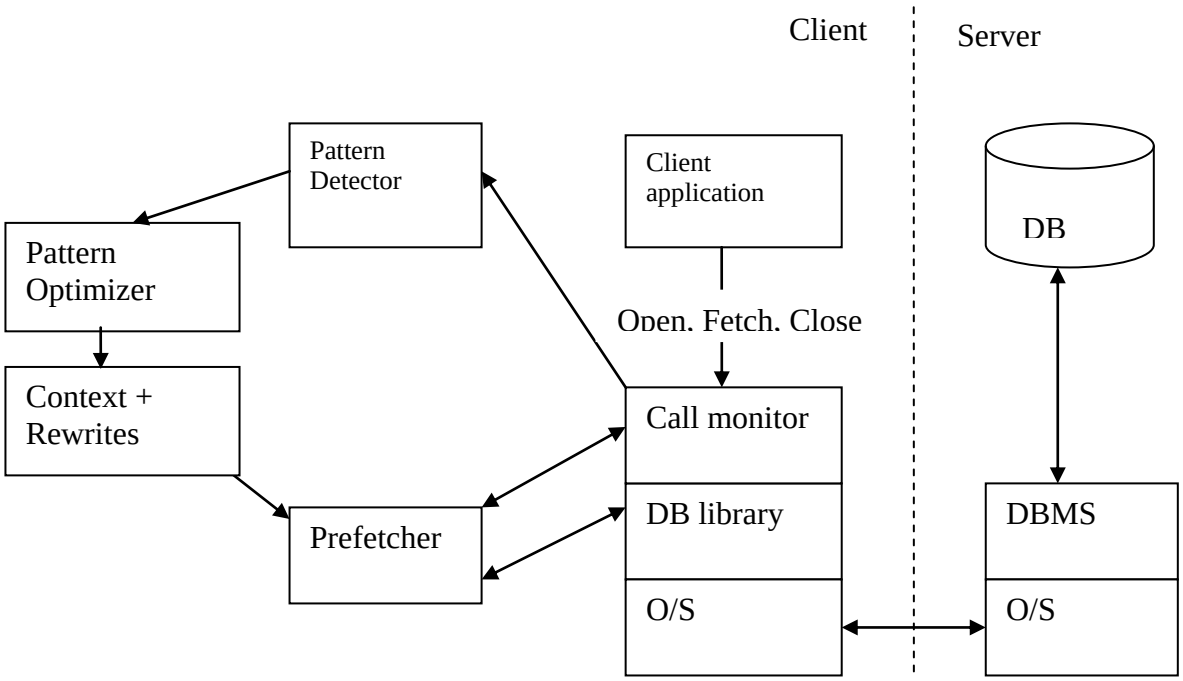

Fig.1: components of the scalpel systems

3. Holistic Optimization by Prefetching Query Results

Ramachandra in [6], [7] proposed a method for semantic prefetching by analysing the control flow and call graph of program binary files. Given the source code for a database application, the system analyses and modifies it, adding prefetch requests into the code as soon as the parameters are known and query execution guaranteed. Since this work is limited to requiring access to the source code of application binaries, it only works for fixed workloads. This model addressed optimizing performance of database/web-service backed applications by means of automatically prefetching

query results, This done by perform prefetching in a calling procedure, even when the actual query is in a called procedure, and not depends on prediction of pattern on access requests.

C. Cache-Based Querying Optimization

Wei Emma in [8] presented a querying system on SPARQL endpoints for knowledge bases that performed queries faster than the state-of-the-art systems. This system features a cache-based optimization scheme to improve querying performance by prefetching and caching the results of predicted potential queries. The SPARQL is a SQL-like structured query language of knowledge bases and represent interfaces that enable users to query publicly accessible knowledge bases.

The cache- based querying optimization addressed main points as : i) it measured the similarity of SPARQL queries by regression model and Euclidean distance, ii) predicted the potential subsequent queries by leverage machine learning technique, iii) prefetched and cached the results of potential queries, and iv) designed a smoothing-based cache replacement algorithm for record-based caching.

Architecture Design for Querying Semantic Knowledge Bases

Fig. 2 shows the layered architecture of the system. The architecture consists of Data;

- *Management Layer:* This layer collects raw sensory data, processes (e.g., filtering and cleaning) the collected data, and transforms the data into meaningful information**.**
- *Knowledge Extraction Layer:* This layer has two main tasks extract knowledge and modeling. It provided two ways in extracting knowledge, namely Curated knowledge extraction obtains facts and relationships from structured data of knowledge corpus and Open Knowledge Extraction (from either arbitrary text available on the Web or real-time sensory data).
- *Knowledge Bases (KBs) and SPARQL Endpoints*: A KB represents facts and relationships associated with the logical assertions of the world.
- *Optimizer:* The Optimizer in the Querying Layer aims to optimize the queries against SPARQL endpoints. It proposed a caching scheme as the optimizer.

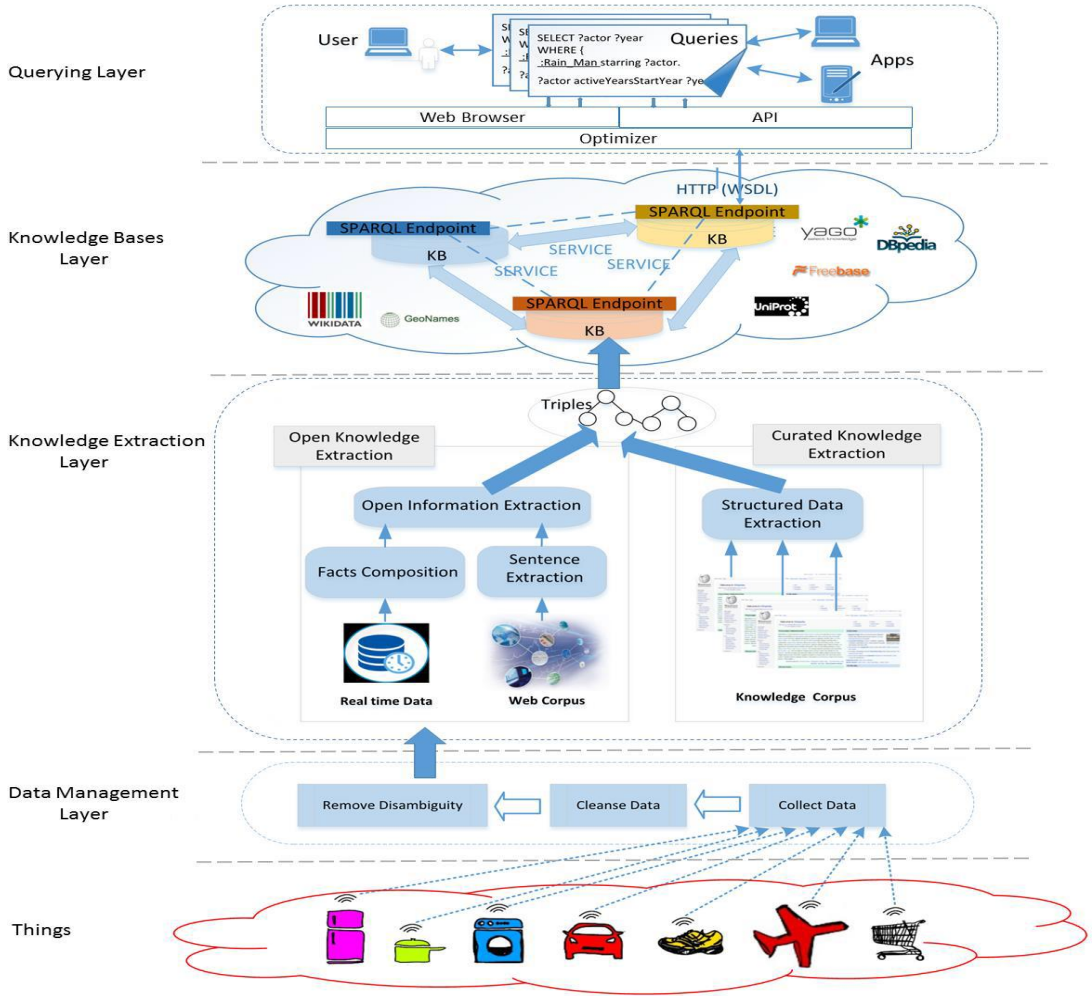

Fig. 2: The Cache- Based System Architecture

D. Apollo system

Apollo is a system layer placed between a client application and the database server. Application clients submit queries to the Apollo system, which then interacts with the database system and cache to return query results as in [9]. This framework in fig.3 learns query patterns and adaptively uses them to predict future queries and cache their results.

From work load, it obtained data access pattern to exploit query relationships in user request. It used query transition graph to learn correlation and it used query pattern aware technique.

The components of Apollo systems

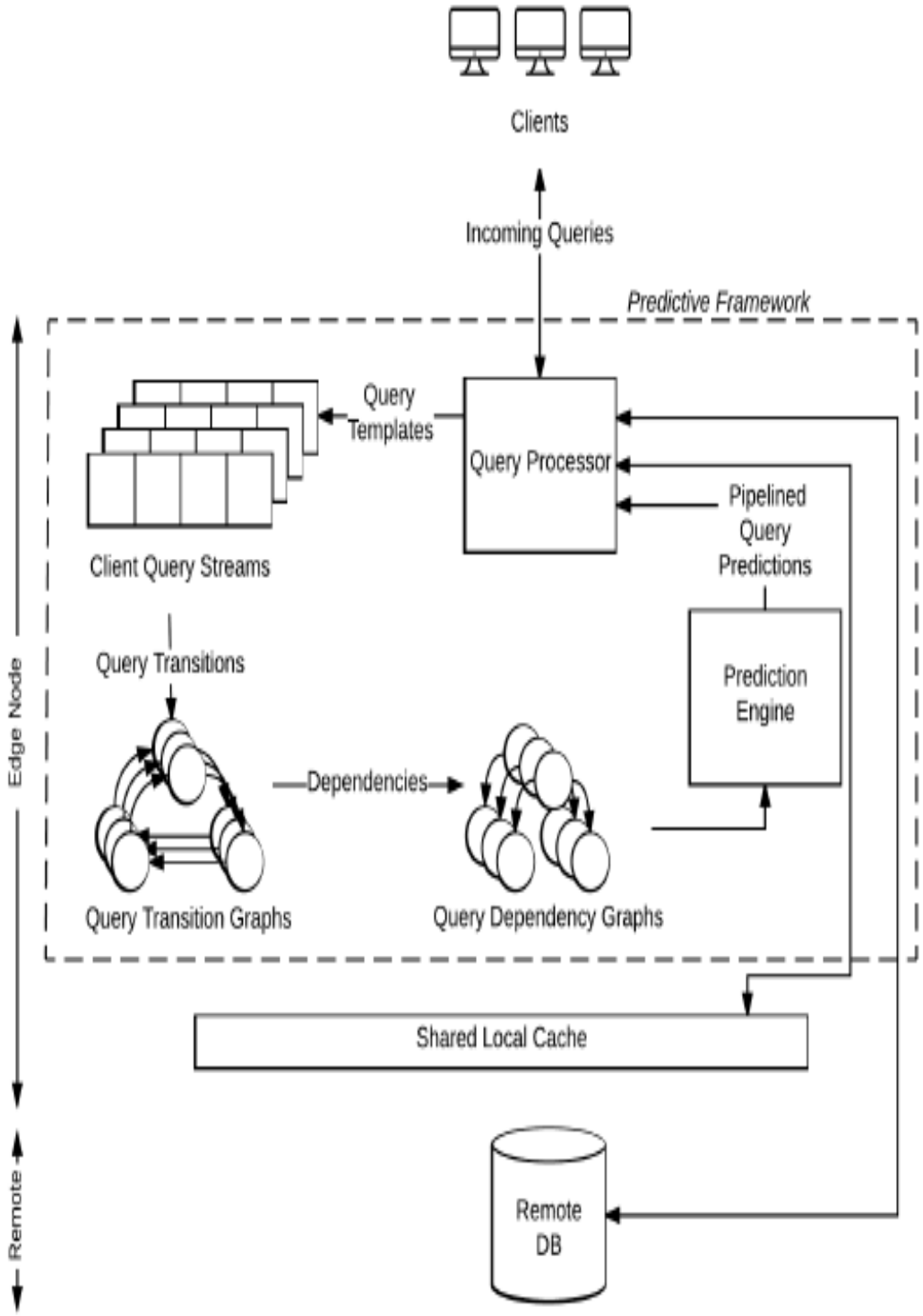

Fig. 3: The Structure of Apollo System

Model used	*Prediction*	*Caching/ prefetching*	*Advantages*	*Disadvantages*
Scalpel system	Pattern detection by greedy heuristic method	Semantic prefetching	• Detect correlation patterns (nesting, batch, and request). • Scalpel detects predictable patterns of queries. • It Minimized response time.	• Make some extra work, when the predicted query hasn't the same of user query. • It is not use all execution strategy such as outer union and merge based rewrite. • Require offline training to detect pattern.
Holistic Optimization	Not used	Prefetch through procedural call	• Increased performance of execution of query • Places prefetch instructions at the earliest possible points while avoiding wasteful prefetches.	• Do not implement a more sophisticated cache manager, • Not predicted with next client request, • Limited to deal with API (Application Program Interface) web service. • Works for fixed workload
Cache-Based Querying Optimization	Leverage machine learning, Euclidean distance, KNN	Used frequency based algorithm, smoothing-based cache replacement algorithm	• Accelerate query execution through using prefeching and caching techniques. • Improved performance and effectiveness of query. • Predicted with the query and suggest similar queries.	• Limited to use large-scale Web data, • It is not deal with the dynamicity of Web data, • It is not investigate of user behaviors that can be used to better predict future queries.
Apollo system	Used query transition graph to learn correlation, and it used query pattern aware technique	Used Memcached, popular industrial-strength distributed caching system,	• It provides significant performance gains over popular, • It provides with caching solutions, • It reduced query response time. • Apollo can scale to high loads by partitioning clients among multiple. • It is able to adapt to changing query patterns over time	

Table1: The search methods in EDS

IV. CONCLUSION

This paper discussed different definition of expert database systems and strategies of integration between two fields (DBMS, AI), also, it discussed systems used to enhance search process in expert database systems through reducing response time. Finally introduced a table that summarize different methods of search through deal with prediction methods used, caching techniques and state strength and weakness points.

REFERENCES

[1] McKay, Donald P., Tim Finin, and Anthony O'Hare. "The intelligent database interface: Integrating AI and database systems." Proceedings of the 1990 National Conference on Artificial Intelligence. 1990.

[2] Kerschberg, Larry. "Expert database systems: Knowledge/data management environments for intelligent information systems." Information Systems 15.1 (1990): 151-160.

[3] Bowman, Ivan T., and Kenneth Salem. "Optimization of query streams using semantic prefetching." ACM Transactions on Database Systems (TODS) 30.4 (2005): 1056-1101.
[4] Bowman, Ivan. "Scalpel: optimizing query streams using semantic prefetching." (2005).
[5] Bowman, Ivan T., and Kenneth Salem. "Semantic prefetching of correlated query sequences." Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. IEEE, 2007.
[6] Ramachandra, Karthik, and S. Sudarshan. "Holistic optimization by prefetching query results." Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. ACM, 2012.
[7] Ramachandra, Karthik, Ravindra Guravannavar, and S. Sudarshan. "Program analysis and transformation for holistic optimization of database applications." Proceedings of the ACM SIGPLAN International Workshop on State of the Art in Java Program analysis. ACM, 2012.
[8] Zhang, Wei Emma, Quan Z. Sheng, and Schahram Dustdar. "A Cache-based Optimizer for Querying Enhanced Knowledge Bases." arXiv preprint arXiv:1807.08461 (2018).
[9] Glasbergen, Brad, et al. "Apollo: Learning Query Correlations for Predictive Caching in Geo-Distributed Systems." (2018).