International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 10 | October - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Big Data Quality: A Systematic Literature Review

Kishan Patel1, Madhvi Bera2

1. B.Tech Student, Department of CSE, Indus University, Ahmedabad, Gujarat, India
2. Assistant Professor, Department of CSE, Indus University, Ahmedabad, Gujarat, India

Abstract:

Big data is the term for large data sets with larger, more varied and complex structures. Difficulties in collecting, analyzing and visualizing for further processes or results. This is the process of exploring large amounts of data to reveal hidden patterns and secrets. A correlation named as big data analytics. One of the most significant problems with Big Data is to gain knowledge through vast amounts of data. Usefulness of extracts Information depends heavily on data quality. In addition to importance, data has quality Recently considered by the big data community and there are none A comprehensive review was conducted in this area. Therefore, this is the main objective of the study to provide a systematic literature review for researchers interested in large Data quality topics. By carefully studying the selected papers, we propose a research tree that divides tasks based on process type, function and technology.

Introduction :

A set of features that frequently co-exist in spatial proximity is called a collocation pattern. Collocation patterns from spatial data have been used in many applications, such as the detection of coexistence of diseases in the spatial proximity of stagnant water or contaminated water bodies.

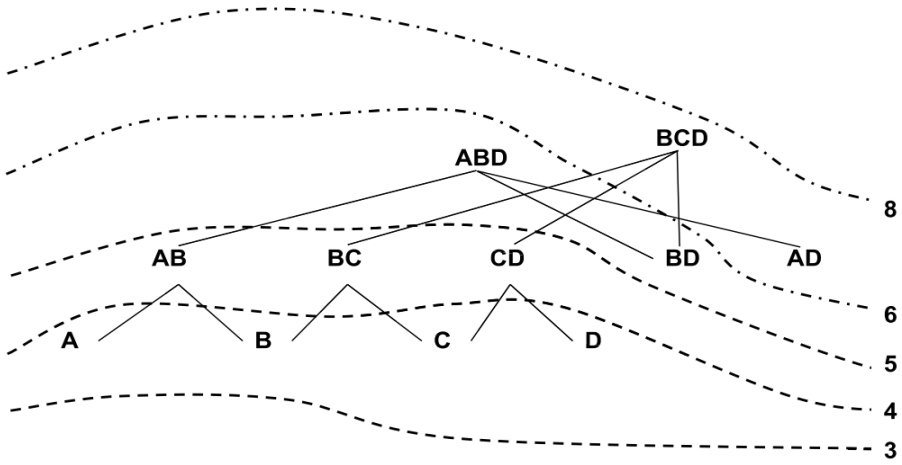


Fig 1. Lattice Visualisation of Collocations Over Range Query D = [3, 8]. [6]

Some of the computational challenges are:

1) There are infinitely many points in a given distance range. How do we find computationally feasible candidate critical distances from these countless choices?
2) Given the enormous search space, how can we efficiently find the distance at which the collocation pattern is no longer valid?, and
3) When mining patterns for candidate critical distances, do we need to recalculate collocations at each critical distance from scratch?, or could there be a better way?

Recalculating the collocation at each candidate distance is computationally expensive and should be avoided. To overcome these challenges, our contributions in this paper are as follows:

1. We introduce a new query type, called range collocation pattern mining, which allows a domain expert to extract collocation patterns over a distance interval, thus facilitating pattern discovery over a large space of neighborhood imagination.
2. We see fig. 1 provides a method for arranging and presenting patterns at spaced intervals as shown in Fig. The presentation method will allow the efficient exploration of spatial collocation patterns and the formulation of relevant hypotheses with ease.
3. We propose an efficient single-pass algorithm for range collocation mining called range − comine that exploits the structural properties of collocation patterns. It also uses a space-efficient data structure to mine patterns efficiently, avoiding multiple runs of the collocation mining algorithm. The memory space requirement of the proposed algorithm is minimal.
4. We conduct extensive experiments on both real and synthetic datasets to show the efficiency of our proposed strategy. Our proposed algorithm outperforms two adapted versions of join-less colocation pattern mining algorithms in terms of space and time requirements. [6]

GeoYCSB microbenchmark :

The GOYCSB microbenchmark includes both data-loading queries and representative geospatial queries, i.e. near, within and intersect. For an apples-to-apples comparison in our experiments, the geospatial queries of MongoDB and CouchBase are written according to a common interface defined in the base type, the GeoDB class. These queries are also written so that a MongoDB query and a related Couchbase query with the same goal return the same result for the same input. This section presents the dataset and workload used for the microbenchmarking experiments conducted in this study, followed by the results of the experiments and our analysis.

The goal of the following experiments is to analyze how efficiently the systems support individual geographic queries for a given workload and system parameters. We also aim to study any performance effects caused by query type and density in a mixed workload setting. We use the same data access distribution for all single node experiments.

Workload A is extended to test topological relationship functions defined by the Open Geospatial Consortium (OGC). This experiment shows that various geospatial tasks can be described in terms of proximity, within, and intersection queries to create a GeoCSB workload.

Workload B In this experiment, we study the impact of geospatial write operations on the systems throughput. The insert operation is selected as the representative write operation. Workload B consists of 80% proximity queries and 20% geospatial insert operations. Both reads and writes are counted to calculate

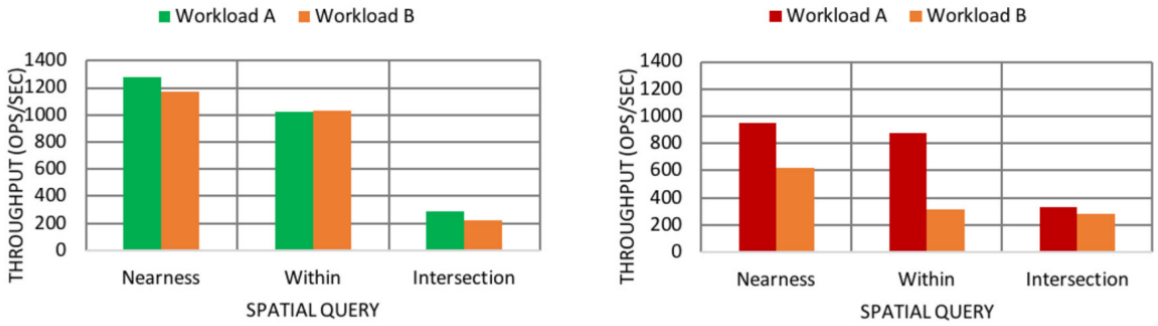
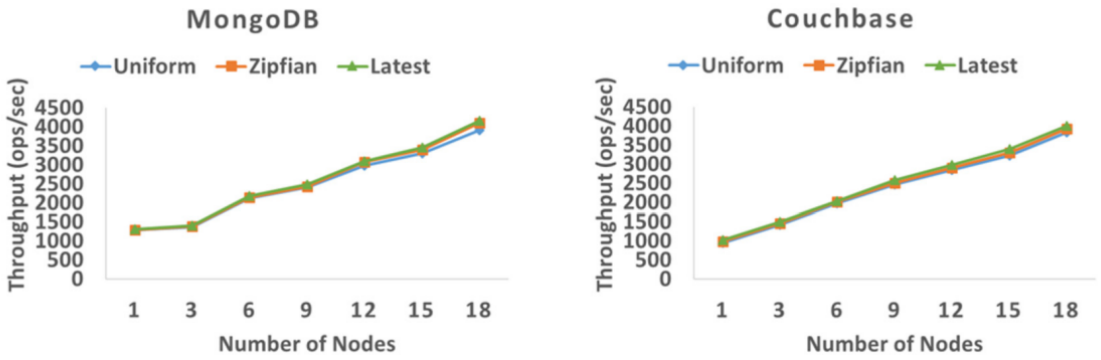


Fig. 4. Throughput Comparisons between Workloads A and B for Individual Geospatial Operations in MongoDB and Couchbase.



throughput.

Fig 2 : Throughputs from MongoDB and Couchbase under Workload A with Various Data Access Distributions.

In MongoDB, throughput measured under workload B does not degrade significantly compared to throughput measured under workload A. However, the throughput degradation under workload B is significant in Couchbase near and within queries. There are several factors to consider for further performance optimization. With 20% insert operations, the database size continues to grow as inserts

continue. Therefore, the sensitivity of FTS-based proximity and database size can be investigated within queries. Also, the overhead of Blew's insertion algorithm can be further investigated.[7]

In recent years, the topic of Social Internet of Things (IoT) has become an emerging IoT field of great interest. IoT is now an established paradigm that supports various applications and services [4]. Social IoT is based on introducing a new vision of cooperation between smart objects similar to what happens to humans when they communicate to achieve a common goal. Therefore, many interconnected IoT devices can be related to each other and provide smart services in different contexts. Sectors that can benefit from this new paradigm and the connection of various smart objects to get increasingly comprehensive and effective answers to their problems are transport, water, energy, etc. The potential for energy efficiency management and consumption forecasting has gradually been recognised. As an important part of sustainable development by governments and energy research institutes. With the increase in population and citizens' living standards, household energy consumption is continuously increasing [4].

The authors adopt a modified k-means clustering algorithm to predict energy consumption in [5]. Algorithms are used to divide users into different groups that are automatically determined based on energy consumption patterns. The authors discuss the findings related to seven clusters of users and make recommendations accordingly. Authors [4] apply three deep learning models to predict household energy consumption which are vanilla LSTM, rank with attention method and rank to rank. In their experiment the vanilla LSTM performed best for predicting energy consumption patterns with a root mean square error metric. These models were not able to predict trends at specific times[5].

Assembling problems in columnar storage :

Column-centered running framework:

To guarantee intensive reads, we need to use the location of different operations in the record assembly. Instead of assembling records one by one, we separate I/O-related operations from the master process, such that a specific thread can be used to intensively read successive blocks of each column. In this section, we present our design goals, and propose a running framework to ensure phased performance based on column order.

Illustrative example. Let's take a query to demonstrate how our running framework runs using an efficient scheduling model. In the example query (in TPCH), we encounter two filters and three fetching columns. The following gives the SQL of the query, where the parentheses give the column abbreviations.

SELECT L.discount (“ld”), L.extendedprice (“le”), L.shipdate (“ls”)

FROM COL WHERE C.nationkey (“cn”) in @SET1

AND L.shipdate (“ls”) BETWEEN “1995-01-01” AND “1996-12-31”

In the example, a select clause composed of three disjunctive filters, saying “cn” in @SET, “ls” ≥ “1995...” and “ls” < “1996...”, could be evaluated before block reads for the projection clause (on “ld”, “le” and “ls”). In the rest of this section, we present a two-staged running framework to parallelize filtering and fetching operations.

Targeting filter-and-aggregate queries, we discovered the I/O degradation derived from small blocks and large-scale parallelism[5]. From micro-benchmarks based on filtering-pushdown, we observe that smaller blocks benefit in reducing the I/O size in typical filtering situations (of which the total selectivity is about 10−3). Our main goal is not only to support efficient filtering-pushdown in columnar storage but also to eliminate I/O degradation in case of small blocks (eg, less than 64 KB). For this, we need to use the I/O location for common filter-and-aggregate use cases in the context of parallel processing[5].

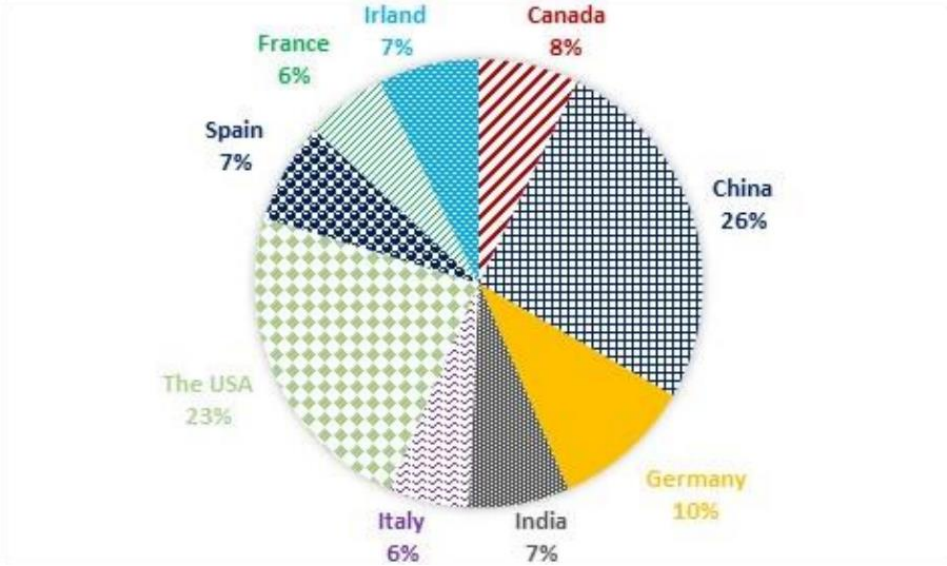
Metadata management. To reduce memory overhead, we provide a read interface representing JSON-like query terms. In our implementation, we only extract metadata for blocks associated with query-declared columns. We found that this change made query initialization easier, especially when configuring smaller block sizes for complex schemas[5]. We can avoid a large amount of unnecessary overhead for wasted blocks.

Bitvector maintenance. The main source of overheads in CORES is maintaining at least one bitset for each query-declared column[5]. Given a complex query, this can result in a dozen bitsets. As we fully evaluate the filtering on each column, the size of the bitvector can be extremely large. Thus, we organize bit vectors with respect to data schema and query patterns. In each level of the schema, we generate only one output bitvector if the column of this level is declared in the query.

Management of thread pool. Our main contribution is to implement an I/O stack using a specific thread pool. We offer method triggers, called through the Reading interface, such as Filter and CreateSchema. Thus, we can transparently support an asynchronous dropout strategy in a filter-and-aggregate query. Based on the thread pool, the IOWorker will asynchronously feed the master (via CQ) by evaluating the

SkippingGrid method. For some heavyweight coding methods, coDec can be used to start decoding threads via user configuration[5].

Results :



Active countries in big data quality scope

Conclusion :

In this paper, we show how to intensively read sequential blocks of each field, and remove invalid payloads based on filtering-pushdown with linear time complexity. Experimental results show that the proposed strategy significantly outperforms its competitors in terms of data compression, especially when queries are composed of high-selectivity filters in analytical workloads. In an ecosystem of smart devices able to communicate with each other to solve common problems, the concept of social IoT was born and developed. In this paradigm, smart objects are autonomous in their communication with others for their needs. We present a common collocation mining problem on distance range queries. To our knowledge, this is the first paper that discusses a computation framework for distance range queries on collocation mining. First, we have discussed a naive approach that gives an a priori work structure of the problem.

In this study, a systematic literature review was conducted in the Big Data Quality era that included three research topics including (a) the type of process, task and technology used in research topics, (b) active researchers, research institutions, countries and places, and (c) challenges and unresolved issues. Therefore, a total of 419 studies were identified, and 170 papers were fully analyzed, and finally, 88 research papers were included in the final paper pool. Also, the information needed to answer the mentioned research

questions was extracted from these studies, and the results show that the quality of big data has been an active and attractive topic in the last decade.

References:

1. A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools by D.P.Acharjya & Kausar Ahmed P
2. RESEARCH ON DATA SCIENCE, DATA ANALYTICS AND BIG DATA By Rahul Reddy Nadikattu
3. Big data analytics: a survey by Chun-Wei Tsai1 , Chin-Feng Lai2 , Han-Chieh Chao1,3,4 and Athanasios V. Vasilakos5*
4. Predicting Household Electric Power Consumption Using Multi-step Time Series with Convolutional LSTM By Lucia Cascone a
5. Accelerating Columnar Storage Based on Asynchronous Skipping Strategy By Wenhai Li
6. Efficiently Mining Colocation Patterns for Range Query By Srikanth Baride
7. GeoYCSB: A Benchmark Framework for the Performance and Scalability Evaluation of Geospatial NoSQL Databases By Suneuy Kim
8. A Review Paper on Big Data Analytics Ankita S. Tiwarkhede1 , Prof. Vinit Kakde2
9. Big Data: A Review Sakshi Saini1 , Amita Dhankkar2 , Suryansh Dabas3
10. Big Data Tools and Techniques: A Roadmap for Predictive Analytics Ritu Ratra, Preeti Gulia
11. Real-Time Big Data Analytics for Data Stream Challenges: An Overview Alaa Abdelraheem Hassan and Tarig Mohammed Hassan
12. Big Data Analytics: A Literature Review Paper Nada Elgendy and Ahmed Elragal