# TERM PAPER
# DATABASE MANAGEMENT SYSTEMS

April 8, 2022

INTERACTIVE DATA EXPLORATION AND VISUALISATION

Roll Number - 19111026
BME/ 6th sem / B.Tech

---

## 1 Introduction

A database management system (or DBMS) is essentially nothing more than a computerized data-keeping system. Users of the system are given facilities to perform several kinds of operations on such a system for either manipulation of the data in the database or the management of the database structure itself. Database Management Systems (DBMSs) are categorized according to their data structures or types.

Mainframe sites tend to use a hierarchical model when the data structure (not data values) of the data needed for an application is relatively static. For example, a Bill of Material (BOM) database structure always has a high level assembly part number, and several levels of components with subcomponents. The structure usually has a component forecast, cost, and pricing data, and so on. The structure of the data for a BOM application rarely changes, and new data elements (not values) are rarely identified. An application normally starts at the top with the assembly part number, and goes down to the detail components.

Hierarchical and relational database systems have common benefits. RDBMS has the additional, significant advantage over the hierarchical DB of being non-navigational. By navigational, we mean that in a hierarchical database, the application programmer must know the structure of the database. The program must contain specific logic to navigate from the root segment to the desired child segments containing the desired attributes or elements. The program must still access the intervening segments, even though they are not needed.

## 2 What is Interactive Data Visualisation ?

Interactive data visualization refers to the use of modern data analysis software that enables users to directly manipulate and explore graphical representations of data. Data

visualization uses visual aids to help analysts efficiently and effectively understand the significance of data. Interactive data visualization software improves upon this concept by incorporating interaction tools that facilitate the modification of the parameters of a data visualization, enabling the user to see more detail, create new insights, generate compelling questions, and capture the full value of the data.

# 3 2 Challenges and Opportunities

Designing a system for interactive data exploration with a human-in-the-loop front-end requires solving a set of very unique research challenges while also opening the door to several interesting opportunities. In this section, we first outline some of the requirements and challenges, followed by an overview of some of the unique opportunities to address them.

Interactive data exploration has a very unique set of requirements (e.g., response time guarantees), many of which are pushing the boundaries of what is feasible today.

## 3.1 Interactive Latencies:

By far, the most important challenge in supporting interactive data exploration is to display a result within the latency requirement. Even small delays of more than 500 ms can significantly impact the data exploration process and the number of insights a user makes. Therefore, a new system for IDE need to maintain certain response time guarantees in order to provide a fluid user experience. Moreover, we believe that a system should be able to refine the query answer progressively. This allows users to get a more accurate answer while visually inspecting the query results.

## 3.2 Conversational Queries:

Different from classical OLAP workloads, users want to explore all different facts of a data set instead of browsing a fixed set of reports. This is very different from what existing analytical databases assume since they expect that the workload is known as prior to create the “right” indexes/samples, whereas the goal of data exploration is to explore and visualize the data in new ways. Moreover, indexes and data cubes suffer from the curse of dimensionality, since memory required is exponential with the number of attributes, making it almost impossible to build an index over all attributes or without knowing the data exploration path ahead of time.

### 3.3 Rare Data Items:

Data exploration often involves examining the tails of a distribution to view the relatively rare data items. For example, real world datasets are rarely perfectly clean and often contain errors (which are typically rare) that can still have a profound effect on the overall results. Similarly, valid outliers and the tails of the distribution are often of particular interest to users when exploring data (e.g., the few billionaires in a dataset, the super users, the top-k customers, the day with the highest traffic). Unfortunately, for rare events and the tail of the distribution, sampling techniques do not work well since they often miss rare items or require a prior knowledge of the workload, a challenge when designing an system for IDE.

### 3.4 Connect and Explore:

Ideally, the user should be able to connect to a dataset and immediately start exploring it. However, this requirement implies that there is no time for data preparation and the system has to build all internal storage structures such as indexes on the fly. Another implication of the connect and explore paradigm is that the system has to stream over larger datasets (from the sources) and may not be able to hold the entire dataset in memory (or even on disk). As outlined in the introduction, online aggregation methods are a good fit to overcome this challenge, since they provide an immediate estimate (with error bars) over the incoming stream. However, online aggregation techniques assume that the data is random, which might be false since some data sources (e.g., data warehouses) often sort the data on some attribute. This can result in a biased estimate of the result and invalid error bars. Similarly, no good estimates are possible if the source returns the data in some chronological order and if there is some (unknown) correlation between time and the value of interest (e.g., the sales are increasing over time).

### 3.5 Quantifying Risk:

An interactive data exploration system with a visual interface allows users to explore hundreds of hypotheses in a very short amount of time. Yet, with every hypothesis test (either in the form of an explicit statistical test or through a more informal visualization), the chance of finding something by chance increases. Additionally, the visual interface can make it easier to overlook other challenges, (e.g., “imbalance of labels” for a classifier) which can lead to incorrect conclusions. Therefore, quantifying the risk is extremely important for an interactive data exploration system.

## 4 OPPORTUNITIES

Although there are several challenges to address, there are many unique opportunities, since data exploration involves close interactions between analysts and the system. Many

of these challenges have not yet been explored within the data management community.

## 4.1 Think Time:

Although the user expects subsecond response times from the system, the system's expectation from the user is different; there might be several seconds (or sometimes even minutes) between user interactions. During this time, the system not only has the chance to improve the current answers on the screen, but also prepare for any future operations. For instance, in our running example, the user might have already dragged out the sex and salary attribute, but not yet linked them together. Given that both attributes are on the screen, the system might begin creating an index for both attributes. Should the user decide to link the two visualizations and use one as a filter, the index is already created to support this operation.

## 4.2 Interaction Times:

Similar to think time, the system can also leverage the user interaction time to provide faster and more accurate answers. For example, it takes several hundred valuable milliseconds to drag an attribute on the interactive whiteboard or to link two visualizations together. In contrast to the previous think time, user interactions are much shorter but usually provide more information to the system about the intent of the user.

## 4.3 Incremental Query Building:

In contrast to one-shot DBMS queries, data exploration is an iterative, session-driven process where a user repeatedly modifies a workflow after examining the results until finally arriving at some desired insight. Think of the session where the user first filtered salary by gender and then added a filter on education. This session-driven discovery paradigm provides a lot of potential to reuse results between each interaction and modification.

## 4.4 Data Source Capabilities:

Traditional analytics systems like Spark and streaming systems like Streambase assume that they connect to a "dumb" data source. However, many data sources are far from "dumb". For instance, commonly the data source is a data warehouse with existing indexes, materialized views, and many other advanced capabilities. While these capabilities do not directly fulfill the needs for interactive data exploration, they can still be used to reducing load and network traffic between the data warehouse and the accelerator. Furthermore, there has been work on leveraging indexes to retrieve random samples from a DBMS. These techniques, together with the possibility to push down user-defined functions (UDFs) to randomize data, provide a feasible solution to the previously mentioned bias problem.

### 4.5 Human Perception:

One of the most interesting opportunities stems from the fact that all results are visualized. Therefore, often precise answers are not needed and approximations suffice. Furthermore, the human eye has limitations and humans are particularly bad at understanding the impact of error bars. The system can exploit both of these properties to provide faster response times (i.e., only compute what is perceived by the user).

### 4.6 Modern Hardware:

Finally, there are several modern hardware trends that can significantly improve the amount of work that can be done in less than 500 ms. While there has been already a lot of work in leveraging GPUs for data exploration, most of the existing solutions focus on single machine setups and ignore the potential of small high-performance clusters. Small high performance clusters can help to significantly increase the amount of available main memory (1–2 TB of main memory is not uncommon with 8 machine cluster), which is crucial for interactive speeds, while avoiding the problems of fault-tolerance and stragglers that come with large cloud deployments. At the same time, fast network interconnects with RDMA capabilities are not only more affordable for smaller clusters, but also offer unique opportunities to decrease latencies. However, taking full advantage of the network requires carefully redesigning the storage layer of the system in order to enable remote direct memory access.

## 5 The IDEA System

The IDEA system is the first system built specifically to enable users to visually explore large datasets through "conversational" interactions. Our prototype addresses many of the previously mentioned challenges (Sect. 2), applying novel progressive sampling, indexing, and query optimization techniques in order to provide interactive response times. In this section, we first provide an overview of our proposed architecture, followed by highlights of research insights and contributions.

### 5.1 Architecture

IDEA's AQP engine is the core of IDEA and divides the memory into three parts: the Result Cache, the Sample Store, and space for Indexes. When triggered by an initial user interaction, IDEA translates it into a query and begins ingesting required data from the various data sources, speculatively performing operations and caching the results in the Result Cache to support possible future interactions. At the same time, IDEA also caches all incoming data in the Sample Store using a compressed row format. When the available memory for the Sample Store is depleted, IDEA starts to update the cache using a reservoir sampling strategy to eventually create a representative sample over the

whole dataset. Furthermore, IDEA might decide to split up the reservoir sample into several stratified subsamples to overrepresent the tails of the distribution, or to create specialized Indexes on the fly to better support visual workloads. All these decisions are constantly optimized based on both past and current user interactions.

## 5.2 Research Findings and Contributions

In this section, we highlight a few selected research findings and contributions of IDEA.

### 5.2.1 Progressive AQP Engine:

IDEA's engine is neither a classical DBMS executionengine nor a streaming engine, instead has an entirely unique semantics. Unlike DBMSs, queries are not one-shot operations that return exact results; rather, data exploration workflows are constructed incrementally, requiring fast response times and progressive results that refine over time. At the same time, streaming engines traditionally deploy predefined queries over infinite data streams, whereas IDEA is meant to enable free-form exploration of data sampled from a deterministic system (e.g., a finite data source).

Fundamentally, IDEA acts as an intelligent, in-memory caching layer that sits in front of the much slower data sources, managing both progressive results and the samples used to compute them. Oftentimes, IDEA has the opportunity to offload pre-filtering and pre-aggregation operations to an underlying data source (e.g., perform a predicate push-down to a DBMS), or even transform the base data by executing a custom UDF in an analytics framework. Finally, in contrast to traditional DBMSs and streaming engines, users compose queries incrementally, therefore resulting in simultaneous visualizations of many component results with varying degrees of error. Maintaining different partial results rather than a single, exact answer imposes a completely new set of challenges for both expressing and optimizing these types of queries. Currently, our IDEA prototype uses a preliminary interaction algebra to define a user's visual queries.

### 5.2.2 Probabilistic Query Formulation:

While developing the AQP enfine of IDEA,we observed that many visualizations rely on the observed frequencies in the underlying data, or estimates of the probability of observing certain data items. For example, a bar chart over a nominal attribute is simply a visualization of the relative frequencies of the possible attribute values (i.e., a probability mass function), and a histogram of a continuous attribute visually approximates the attribute's distribution (i.e., a probability density function). Although seemingly trivial, this observation prompted us to reconsider online aggregation as a series of probability expressions. This novel probability formulation actually permits a wide range of interesting optimizations including taking advantage of the Bayes' theorem to maximize the reuse of results. Our current implementation of IDEA therefore manages a cache

of results that stores previously computed frequencies and error estimates for reuse in future queries.

### 5.2.3 Visual Indexes:

Similar to the algebra and optimizer, we also found that traditional indexes are not optimal for interactive data exploration tasks. Most importantly, existing techniques either sort the data (e.g., database cracking) or do not naturally support summary visualizations. As previously mentioned, sorting can destroy data randomness and, consequently, the ability to provide good estimates. Similarly, indexes generally index every tuple without considering any properties of the frontend (e.g., human perception limitations, visualization characteristics). This approach often results in very large indexes, especially with increasingly large samples or highly dimensional data.

### 5.2.4 Sample Store:

As previously mentioned, IDEA caches as much data as possible from the underlying data sources in order to provide faster approximate results, since most data sources are significantly slower. For example, the memory bandwidth of modern hardware ranges from 40–50 GB/s per socket, whereas we recently measured that PostgreSQL and a commercial DBMS can only export 40–120 MB/s, even with a warm cache holding all data in memory. Although DBMS export rates may improve in the future, IDEA's cache will still remain crucial for providing approximate answers to visual queries and supporting more complex analytics tasks (e.g., ML algorithms).

### 5.2.5 Inconsistencies:

Interactive response times often require computing approximate answers in parallel, which can lead to inconsistencies in concurrent views. Similarly, an outlier that appears in one result visualization may not yet be reflected in another, causing the user to draw a potentially incorrect conclusion.

Although initially assuming that inconsistencies would pose an important challenge for IDEA, we found that this problem only arises in a few corner cases, and we did not observe any consistency issues during various user studies. In particular, IDEA's result reuse and sampling techniques work together to mitigate many potential consistency problems, and any noticeable differences tend to disappear before the user can even recognize them.

# 6 Conclusion

In this paper, we presented the case for a new bread of data management systems which seek to maximize human productivity by allowing users to rapidly gain insights from new

large datasets. We outlined the research challenges and opportunities when building such a new system and discussed the insights we gained from building our system called IDEA. Finally, we discussed other important considerations in the context of building interactive data exploration systems including benchmarking, natural language interfaces, as well as interactive machine learning.