A Structured Review of Data Management Technology for Interactive Visualization and Analysis

Leilani Battle and Carlos Scheidegger

Abstract— In the last two decades, interactive visualization and analysis have become a central tool in data-driven decision making. Concurrently to the contributions in data visualization, research in data management has produced technology that directly benefits interactive analysis. Here, we contribute a systematic review of 30 years of work in this adjacent field, and highlight techniques and principles we believe to be underappreciated in visualization work. We structure our review along two axes. First, we use task taxonomies from the visualization literature to structure the space of interactions in usual systems. Second, we created a categorization of data management work that strikes a balance between specificity and generality. Concretely, we contribute a characterization of 131 research papers along these two axes. We find that five notions in data management venues fit interactive visualization systems well: *materialized views*, *approximate query processing*, *user modeling and query prediction*, *multi-query optimization*, *lineage techniques*, and *indexing techniques*. In addition, we find a preponderance of work in materialized views and approximate query processing, most targeting a limited subset of the interaction tasks in the taxonomy we used. This suggests natural avenues of future research both in visualization and data management. Our categorization both changes how we visualization researchers design and build our systems, and highlights where future work is necessary.

1 Introduction

When building systems for interactive data analysis and visualization, the most important question we keep in mind is: "will this system actually help its user in what they are seeking to achieve?" The mismatch between *system-as-actually-built* and *system-as-user-wish-it-were* is a common way for things to wrong, and is now generally regarded as one of the main threats to validity in visualization design [123]. This shift from "how can we build visualization systems?" to "what visualization systems ought we build?" is one of the most important perspective changes in visualization research.

One area of particular interest is the shift towards interactive visualization of big data. The success our community has experienced over the years means we now want to build visualization systems for increasingly large and more complex datasets. However, a fundamental assumption made in many of our visualization techniques is that the input data will always be small enough to process and manage ourselves, enabling us to sidestep questions regarding scalability and computational efficiency. For sufficiently large datasets, we can no longer afford to ignore computational constraints. As system builders and researchers, we seek technologies that enable us to respect our users and their particular constraints, while increasing the range of dataset size and complexity that our systems can handle.

We are left, then, with a conflict between **computational constraints from scale-centric concerns** and **design constraints from user-centric concerns**, which requires combining lessons from the visualization and data management communities. Through a structured review methodology, we argue that tools appropriate for the user-centric concerns which also gracefully handle scale-centric concerns are not all at the same level of maturity. Some limited areas are ready to support the development of actual systems, some areas still require the development of methods and technical solutions, and some areas require foundational work. To support these claims, we contribute:

- a **structured literature search and review** of the last three decades of publications in top academic venues in visualization, databases and data management, specifically targeted towards interactive visualization of large datasets,
- a **two-axis categorization of the found articles**: task taxonomies designed for interactive visualization versus different optimization strategies,
- a **discussion of observed gaps** in the literature and other patterns we have found, and
- a **reflection on limitations of our methodology**, and where future work is needed

Our primary audience is graduate students, researchers, and practitioners in visualization. Specifically, we target those who have struggled with the implementation of interactive analysis or visualization systems when data scale becomes a problem. This issue typically manifests as application performance issues, but correctness problems can come into play as well. In response, the technologies we review here are positioned to make a positive impact on the ways future visualization system builders design their solutions.

1.1 Three ways to develop data visualization programs

Consider the following scenarios, which capture how interactive visualization programs are designed and implemented. There are two axes of interest: how data is accessed, and how visualizations are built from the data.

Data access: connect to a DBMS directly The standard way to access data from a source of interest is to have the visualization program connect directly to a traditional database management system (DBMS) storing the data, using a language like SQL. This provides maximum flexibility for the application and DBMS: they do not depend on one another's computational characteristics at all. Unfortunately, in this case the DBMS lacks any application context that could be used to further improve system performance, such as visual encodings, interactive filtering, and so on. As we will show later in this review, modern data management literature has started to provide solutions for some of the problems in interactive visualization, but most DBMSs lack these necessary features to support visualization programmers. As a result, flexibility currently limits scalability.

Visualization creation: use a modern visualization DSL After the problem of data access has been addressed, visualization programmers need to design the actual visualization aspects of their applications. When designing for flexibility, programmers often implement their desired interactive data visualization functionality using modern domain-specific languages (DSLs) for visualization, such as Vega or VisQL [153, 154, 165]. These DSLs capture both visual encodings and interactive behavior in a machine-readable form, which would ideally

- *Leilani Battle is with the Department of Computer Science at University of Maryland. email: leilani@cs.umd.edu*
- *Carlos Scheidegger is with the HDC Lab at the Department of Computer Science, University of Arizona. email: cscheid@email.arizona.edu*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

be translated into a form that communicates to the DBMS the computational and perceptual features of the visualization, so interactions and data access can be optimized in tandem. Unfortunately, this vision is unrealistic today. Libraries like Vega are flexible and powerful, but do not currently provide the infrastructure necessary to communicate information to the DBMS with the goal of optimizing the performance of the downstream queries, though we expect this situation to improve in the next decade or so. Once again, flexibility seems to limit scalability.

Roll up your sleeves, write from scratch If programmers today want to design effective, interactive visualizations that scale, the current situation requires them to know about recent techniques in the data management research literature. We believe this problem makes reviews such as ours uniquely important: data scale has caught up with the field and our high-level infrastructure is not well-suited for big data contexts. In addition, a long-term solution to this problem will require interactive visualization design to happen within the bounds of computational constraints. We anticipate that designers of novel visual encodings and interaction modalities with a good working knowledge data management techniques will have a unique advantage, since their novel encodings would ideally be optimized by (future) DBMSs.

2 Related Work

We are not the first to observe the necessity of connecting data management to interactive visualization research. A number of other surveys have sought to bridge the gap between the two fields.

We highlight here two classes of reviews: those written from the perspective of visualization research, and those written from the perspective of data management research. In the former, Bach et al.'s review of temporal data visualizations [7] organizes visualization work along data access requirements; Godfrey et al. [62] provide a general overview of work handling scale concerns in data visualization; and Zhang et al. [190] provide a review of commercially-available systems for big data visual analytics (as of 2012).

In data management, we note the following surveys: Chaudhuri and Dayal [31] provide a classic overview of data cubes and related work; Halevy [66] surveys papers on *views*: the more general idea of answering a query using previously computed tables derived from the original database; Mami and Bellahsene provide a survey on the problem of *which views to compute* [110]; Li and Li [100] survey approximate query processing, the idea of allowing a small amount of error into the result in order to provide answers more efficiently; Kwon et al. provide a visualization-focused survey in the same area [96].

Of these surveys, we note that only Kwon et al. focus on connecting one specific database technique — sampling — to interactive visualization opportunities. Our work is similar in that respect. In contrast to the relatively narrow focus in sampling, we offer a broader survey of both visualization and data management work. In addition, we categorize the surveyed work by applicability to particular interaction tasks as well as by the relevant optimization area in data management research. This categorization is, to the best of our knowledge, unique to our work. We describe our methodology in the next section.

3 The Survey and its Methodology

Our survey includes a total of 130 references selected from a set of 278 research articles from top data management, visualization, and human-computer interaction conferences: SIGMOD, VLDB, ICDE, IEEE VIS, Eurovis, UIST, and ACM CHI. We obtained the larger set by identifying relevant keywords in notable papers, and then performing searches on IEEE Xplore, ACM Digital Library, and Google Scholar. Our goal in this review is not to be comprehensive. Rather, we keep to a relatively narrow search to provide a view into the relative development of subareas, giving context and structure for future visualization students, researchers, and practitioners. For interested readers, we provide the full bibliography as supplementary material.

3.1 Paper Selection

We selected papers for review as follows:

1. We reviewed the proceedings of prominent visualization and HCI venues (VIS, CHI, UIST, EuroVis, etc.) from 2000 to 2020. We selected papers that used existing database techniques with a focus on improving performance.
2. We clustered the collected visualization and HCI papers by the database optimization techniques they applied (e.g., materialized views, user modeling, etc.).
3. We then reviewed the database literature for influential papers related to the topic clusters derived from Step 2.
   (a) Database papers that were directly referenced by the papers from Step 2 were included for review.
   (b) We then conducted a formal literature search of the last 20 years of data management research by inspecting the titles and abstracts of every paper in this time frame that was published at ACM SIGMOD, VLDB, and ICDE, arguably the top three data management conferences in the field; any relevant and influential papers were included in our analysis.
   (c) Finally, we performed online searches for each topic (e.g., via google scholar), and included any influential papers missing from our current list (i.e., papers referenced by many other papers in the database community).
4. As a verification step, we asked colleagues from the database community to review our list of topics and corresponding papers; we included any papers considered relevant to the given topics but missing from our review list, (e.g., adding the "interactive" keyword to our database search based on peer feedback).

Our paper selection process yielded 72 visualization and HCI papers, 186 database papers, and 20 other surveys for review. We prioritize relatively recent full and short papers. However, a small number of workshop papers are included, as well as papers published before 2000. We did so when they appeared as a top result in our online searches.

Keywords Our search keywords for the visualization literature are slanted towards databases and systems, emphasizing data management operations and optimizations: "aggregat", "sampl", "quer", "cub", "large", "scal", "multi", and "big". Conversely, our keywords for the database literature focused on visualization and HCI-oriented topics: "interfac", "interact", "visual", "explor", and "real-time".

Topics Omitted from Our Review We focus on techniques to improve the performance of online analytical processing (or OLAP) use cases at interactive speeds. Given the strong emphasis in the database literature on managing structured (i.e., tabular) data in OLAP use cases, we focus on optimizations for tabular data in this work. As a result, certain topics are omitted from our analysis, particularly techniques for interacting with large graphs, image collections, and text corpora (i.e., unstructured and semi-structured datasets), and methods for visualizing massive simulations. These areas are treated as special cases within databases, and often require additional data management techniques that are outside the scope of this work. We also adopt the common convention in the (database) OLAP literature to focus on techniques that address scale-up in the *number of data points*, and not necessarily the dimensionality of the data. As such, dimensionality-focused techniques are outside of the scope of this work. Finally, we point out that there is an extensive body of work in scientific visualization to support big data use cases in the physical and biological sciences (e.g., volume and flow rendering, and visualizing output from massive simulations). These use cases generally do not match the properties of OLAP, and so are out of scope for this survey.

3.2 Coding Methodology

For each paper selected for review, we labeled the papers by both: 1) database optimization(s) applied and 2) interaction type(s) supported. Note that a single paper can have multiple optimization and interaction labels, and thus may appear multiple times in our analysis. We apply the database optimization labels derived from our paper clustering. We use an a priori coding strategy for the interaction type labels using the types specified in the Brehmer and Munzner multi-level typology [20]. We detail our interaction coding strategy here, and describe our derived optimization clusters in Section 4.

3.2.1 Choosing an Appropriate Interaction Taxonomy

To reiterate, a primary threat to the validity of visualization and analysis systems is that of *solving the wrong problem* [123]. As a result, a significant amount of visualization methodology research seeks to characterize more precisely what it is that users want to do when they use visualization systems (we assume, as the literature does, that this characterization is possible in principle). We tailor our survey to visualization researchers and practitioners, and directly relate our surveyed data management research to methodological work in data visualization. Concretely, we place the surveyed data management work inside the taxonomies that characterize analytical tasks in visualization.

To identify an appropriate taxonomy, we searched the visualization literature for "taxonom", and identified three candidate taxonomies that apply in our setting: those of Amar, Eagan and Stasko [5], Brehmer and Munzner [20], and Heer and Shneiderman [70]. After simplifying their respective structures to a list of analytic tasks, we aligned tasks across the three taxonomies, arriving at the following task categorization: `encode`, `select`, `navigate`, `arrange`, `change`, `filter`, `aggregate`, `annotate`, `derive`, `record`, `import`. Note that here, `encode` refers to interactions that users perform to create new visualizations, and not the rendering process itself.

In our view, Brehmer and Munzner's typology best captures distinctions in how data management technology can be applied in interactive analysis systems, balancing abstraction in how tasks are carried out, and concreteness in which tasks users actually seek to achieve. We focus on the "how" level of their typology, because it provides a rich structure for classifying interactions by behavior. The "how" level of the typology emphasizes methods for "`encoding` data", "`manipulating` existing elements in a visualization" and "`introducing` new elements" [20]. These classes have clear translations to data processing operations. For example in the `manipulate` class, the `filter`, `aggregate` and `derive` methods directly translate to specific SQL clauses (`WHERE`, `GROUP BY` and `SELECT`, respectively).

3.2.2 Coding by DBMS Optimization Techniques

We adopted a descriptive coding process to cluster papers by DBMS optimization techniques [150]. After the initial set of papers were identified from Step 2 of our paper selection process, two authors independently designed descriptive codes to categorize the core topic(s) for the observed optimization techniques. The coders then discussed any disagreements, and iteratively refined the codes to form a final codebook. As an example, while `materialized-views` was identified in the first iteration, `index` was identified on subsequent discussions.

3.2.3 Coding Papers

In this work, we are specifically interested in how database optimizations relate to specific interaction types. When determining if a particular database paper supports a given interaction type, we look to our list of reviewed visualization and HCI papers for guidance: if a visualization or HCI project uses an optimization technique to support a particular interaction, then we know that it is possible to apply the corresponding interaction type label to papers that discuss this optimization strategy. For example, not all papers on data cubes develop visualization interfaces to test the techniques, but data cubes papers in the visualization community (e.g., imMens and Nanocubes [104, 107]) are primarily designed to support `navigation` and `filter` interactions, thus other data cubes papers that shared similarities with imMens and Nanocubes were also assigned these interaction labels.

Not all database implementations provide support for all interactions. For example, naive sampling strategies may not actually support sampling over queries involving filters or complex transformations, which are required to support the corresponding `filter` and `derive` interaction types. Thus the interaction labels applied to our selected papers often represent the *maximal set* of interaction types supported by the relevant database optimization techniques. For example, the Pangloss system [118] uses approximate query processing (or AQP) optimization techniques to support `aggregate`, `derive`, `filter`, `navigate`, and `record` interactions over large datasets. However, we find that most proposed database optimization strategies for AQP are designed primarily to only support `aggregate` and `filter` interactions.

4 An Overview of Database Optimization Techniques

Our analysis led to a clustering of papers by the database optimization strategies used to improve system performance. In this section, we describe each database optimization type produced in our clustering, explain the significance of this optimization, and provide examples of how it is or may be applied in visualization contexts.

4.1 Materialized Views

What is it In a classic database view, the underlying query is always (re-)executed every time a client (e.g., application, database user) accesses the view, and the result is discarded after the client is done accessing the view. A *materialized* view is simply a view where rather than discarding the result, the DBMS stores it as (e.g.) a new table in the database. Materialized views shine in their re-use: the more frequently a view is accessed, the greater performance benefits are provided by materializing it. On the other hand, if the underlying data changes often, the materialized view queries generally have to be updated to reflect the new results.

Data cubes are a special case of materialized views to better support Online Analytical Processing (or OLAP) workloads, where aggregate queries are quite common. Data cubes represent the materialization of multiple aggregation queries. Given a sorted list of grouping attributes G and separate attributes from which to compute aggregate statistics A, data cubes materialize the results of all queries representing the calculation of all aggregates $a \in A$, *after* grouping the data using some prefix of the grouping attributes G' where $|G'| \leq |G|$. Each grouping attribute $g \in G'$ represents a dimension of the cube.

Why this matters for Vis As seen in previous visualization work [8, 104, 107], materialized views can provide two clear performance benefits: 1) little to no computation needs to be performed at runtime, ensuring that interactions can remain interactive, even on large datasets; and 2) when using aggressive aggregation strategies, the aggregate results can be compressed into significantly smaller space compared to the raw data, making the data more manageable for end users. However, materialization is not a panacea and depends on the computation being performed. In particular, the size of materialized results tends to be a limiting factor, as they can quickly become unwieldy if left unchecked [104, 119, 129]. For example, it is unlikely that the aggregate results for all possible interactions should be pre-computed, as they could produce results that are many times the size of the original dataset (e.g., as found by Moritz et al. and Battle [15, 119]). Fortunately, this limitation has been considered extensively in the database literature, providing us with an opportunity to leverage materialized results more effectively for visualization use cases [15, 37].

Example System Two prototypical examples are imMens by Liu and Heer, and Nanocubes by Lins et al. [104, 107]. These systems precompute data cube-like structures in advance, before a user performs any interactions. These structures enable fast indexing into the data to identify which tuples are relevant to the current viewport (e.g., after a pan, zoom, or selection interaction) and retrieval of the pre-aggregated results relevant to the viewport with low latency.

4.2 Approximate Query Processing

What is it When working with large datasets, oftentimes users can quickly gain a sense of important trends and salient features with just a small randomized sample of the data. This key idea has lead to a deep interest in the database community in supporting "approximate query processing" (or AQP), which specializes in executing database queries using randomized samples of the underlying data to give users fast (and in some cases, near-immediate) feedback. However by definition, approximate results have inherent error, which these approximate systems aim to calculate, bound, and ultimately minimize.

Approximate query results are primarily computed using statistical samples, for example using progressive algorithms, where the results are initially computed using a small sample of the data, then tuples are continuously added to this sample and the results are recomputed to improve the results over time. The approach of updating results in real time, and allowing users to stop when they have "seen enough" [143] is generally referred to as *online* analytics in databases and *progressive*

	materialized-views	aqp	mqo	user-modeling	provenance	index
encode			[48]	[48] [90] [109] [180]	[18]	[80] [185] [189]
select	[13] [34] [47] [83] [87] [95] [112] [116] [139] [168] [169] [170] [177]	[34] [87] [99] [138]	[47] [61] [76] [83] [91] [116] [140] [149] [158] [170] [195]	[33] [51] [79] [127] [193]	[13] [139]	[16] [26] [34] [53] [79] [112] [139] [157] [168] [169] [193]
navigate	[8] [22] [35] [37] [39] [44] [52] [58] [64] [68] [74] [82] [88] [89] [95] [101] [102] [103] [104] [105] [107] [112] [113] [114] [115] [122] [124] [129] [132] [134] [136] [145] [148] [147] [151] [152] [162] [164] [166] [168] [169] [170] [174]	[38] [39] [41] [44] [50] [71] [72] [84] [88] [89] [93] [118] [132] [136] [145] [151] [161] [173] [186]	[48] [136] [170]	[8] [43] [46] [48] [78] [79] [88] [89] [105] [151] [152] [161] [164] [28] [193]	[167]	[16] [22] [44] [45] [50] [53] [59] [75] [79] [80] [93] [101] [112] [129] [134] [157] [168] [169] [173] [185] [186] [189] [193]
arrange	[39] [159] [170] [172]	[39] [92] [144] [159] [172]	[60] [170]			
change			[48]	[48] [109]	[18]	[80] [185] [189]
filter	[1] [2] [3] [6] [8] [13] [19] [22] [27] [29] [30] [35] [36] [37] [39] [44] [47] [52] [56] [57] [58] [64] [68] [74] [82] [83] [87] [88] [89] [95] [101] [103] [104] [105] [107] [112] [114] [115] [116] [117] [119] [124] [125] [129] [134] [135] [136] [139] [145] [147] [148] [151] [152] [162] [164] [166] [168] [169] [170] [172] [174] [177]	[1] [2] [3] [4] [6] [30] [29] [36] [39] [41] [44] [50] [54] [56] [57] [71] [72] [84] [85] [86] [87] [88] [89] [93] [99] [98] [119] [118] [126] [135] [136] [138] [145] [151] [172] [173] [186] [188] [191]	[47] [56] [61] [60] [76] [83] [91] [116] [136] [140] [149] [158] [170] [171] [195]	[8] [33] [43] [46] [51] [78] [79] [88] [89] [105] [127] [151] [152] [164] [28] [193]	[13] [139] [167]	[16] [22] [26] [36] [44] [45] [50] [53] [56] [59] [75] [79] [80] [93] [101] [112] [117] [125] [129] [134] [139] [157] [168] [169] [173] [185] [186] [189] [193]
aggregate	[1] [2] [3] [6] [8] [19] [27] [29] [30] [34] [35] [36] [37] [39] [44] [47] [52] [56] [57] [58] [64] [68] [74] [82] [83] [88] [89] [95] [101] [102] [103] [104] [105] [107] [112] [113] [114] [115] [116] [117] [119] [122] [124] [129] [134] [135] [136] [139] [145] [147] [148] [151] [152] [159] [162] [164] [166] [170] [172] [174] [175] [177] [187]	[2] [1] [3] [4] [6] [25] [29] [30] [34] [36] [39] [44] [50] [54] [56] [57] [71] [72] [85] [86] [88] [89] [93] [92] [99] [98] [119] [118] [126] [130] [135] [136] [138] [145] [151] [159] [172] [173] [176] [186] [187] [188]	[25] [47] [48] [56] [60] [76] [83] [91] [116] [136] [149] [170] [171] [175]	[8] [48] [88] [89] [90] [105] [109] [151] [152] [164]	[18] [139]	[34] [36] [44] [50] [53] [56] [80] [93] [101] [112] [117] [129] [134] [139] [173] [185] [186] [189]
annotate						
derive	[2] [30] [44] [47] [56] [83] [116] [134] [170] [175]	[2] [25] [30] [44] [56] [99] [118] [126]	[25] [47] [48] [56] [60] [83] [91] [116] [149] [170] [175]	[48]		[44] [56] [80] [134] [185] [189]
record		[118]				
import						[80] [134] [185] [189]

Fig. 1. Our literature search resulted in 130 total references coded by implementation strategy and interaction type, and is summarized in this table. Each subarea, represented by a cell, is colored based on the "maturity level" of the specific intersection of the research areas: fewer than 10 references, 10 to 19 references, and 20 or more references. As additional material, we provide a separate BibTeX file containing all of these references and the coding metadata. We have kept the row "annotate" unpopulated since we could not identify any references in our search which would fit the context of this review. We provide more discussion about our results in Section 5.

analytics in visualization. Data samples can also be pre-computed offline, for example by creating a stratified 5% sample of the original data, grouped by the most popular attributes. While statistical sampling is the most popular approach, there are alternatives, such as early termination: strategically stopping a scan of a table before it is finished can be an effective way to achieve approximate results [24, 94].

Why this matters for Vis Admittedly, several existing projects in visualization already do a great job of motivating (albeit naive) approximate and progressive data computation strategies (e.g., Pangloss [118]). However, basic sampling only goes so far [3], and quickly loses its effectiveness at massive (i.e., terabyte and petabyte) scales, which are the every-day reality of many high-impact companies, such as Google, Facebook, and Microsoft. Considering more sophisticated and statistically-aware sampling strategies from the database community can help us to extend the reach of visualization work. In addition, the additional information provided by a concrete visualization scenario might inform some of the decisions necessary for achieving AQP systems (such as the choice of stratification in systems like BlinkDB [3]).

Example System Pangloss, developed by Moritz et al. [118], enables users to "explore very large datasets by grouping, aggregating, and filtering," where these data processing operations are computed and rendered iteratively over time. Users can also direct Pangloss to continue processing certain visualizations in the background, so that users can revisit an approximate result later on to verify that the final results match previous observations.

4.3 User Modeling & Query Prediction

What is it Exploratory visual analysis has been shown to incorporate consistent [184], predictable (e.g., [21, 65]) data analysis tasks and interaction sub-sequences. Specifically, one can derive models of user behavior that can predict user personality characteristics [21], future interactions or data fetches [8, 28, 46, 88], or even whether the user is close to extracting insights or completing a task [9, 65]. These models can be derived using rule-based methods [28, 46, 65] or automatically using machine learning techniques [8, 21].

In general, the database community has been interested in how a series of data requests can be detected, anticipated and managed more efficiently over the course of one or more analysis sessions. Query prediction is but one part of this user modeling process, where the system anticipates future requests. Another issue is handling requests once they have been received, where it is possible that not all requests can be processed in time for the user to see them. As such, we can apply user modeling techniques to prioritize existing requests to further reduce system latency.

Why this matters for Vis. Though user modeling has been explored in the visualization community (e.g., [21, 63, 65]), these models are not straightforward to incorporate into existing systems. For example, many models are derived by hand, making it difficult to map the results to new use cases or improved system designs. More can be done to make these models more concrete, for example, machine learning techniques have shown promise in automating the modeling process (e.g., [8, 21]). With encoded models comes opportunities to automate system performance optimizations, such as automated data pre-fetching and speculative or predictive data processing [8, 88].

Inconsistent data collection and sharing methodologies impacts our ability as a community to power data-driven user modeling systems [11]. A community effort to collect and share data through a consistent process could empower the community to develop more robust algorithms and theoretical models for user visualization and interaction behavior, as well as enable broader evaluation of our research output and impact (e.g., through meta-analyses [81, 97]).

Example System Battle et al. developed the ForeCache and Sculpin systems to support interactive browsing of massive array data via pans and zooms [8, 10]. The main contribution of the work by Battle et al. is a multi-level prediction engine that anticipates future pans and zooms, and pre-fetches the corresponding data ahead of users as they explore. By pre-fetching data in advance, the system appears to respond faster to the user's interactions, thereby improving the user's data browsing experience.

4.4 Multi-Query Optimization (MQO) & Shared Work

What is it When multiple queries will be executed in parallel (i.e., together in a batch), most database query optimizers will select an optimized execution plan for each query independently. However, in scenarios where either the same operations (i.e., shared work [61, 140, 195]) or same data (i.e., shared intermediate results [83, 149, 158]) appear in multiple queries in the batch, the query optimizer could develop a smarter global execution plan that exploits these overlaps. The concept of global optimization across multiple parallel queries is formally referred to as "Multiple-query optimization" or "multi-query optimization". Multi-query optimization is known to be a problem of exponential complexity, which worsens considerably as the number and

complexity of queries to optimize increases [149]. To address this problem, most solutions focus on aggressively pruning the search space of possible plans, for example by optimizing queries independently and then merging the resulting plans; or by optimizing the query plans one at a time, seeing how each subsequent plan can reuse existing plans [149]. Others have also considered probabilistic methods [83]. All methods utilize tree or DAG structures to represent query plans [61, 83] and often also utilize aggressive tree pruning techniques to further reduce the search space for shared plans.

Why this matters for Vis Multi-query optimization could be particularly interesting, given the proliferation of batch-based visualization techniques like visualization recommendation [40, 77, 171, 181]). When enqueueing tens or hundreds of recommended visualizations in response to a single user interaction, it may be advantageous to consider how the corresponding data processing operations (e.g., DBMS queries [165]) could be merged for more efficient processing over massive datasets. In addition, we note that multiple-coordinated-view (MCV) systems are natural targets for MQO: almost by definition, each view can be modeled as a separate query over the same database.

Example System Consider the SeeDB system developed by Vartak et al., which aims to recommend new visualizations for users to explore [171]. SeeDB's recommendation strategy is to identify interesting statistical differences between subsets of data that may be of interest to users, such as differences in capital gains between married and unmarried individuals. To produce a recommended visualization, SeeDB must execute the corresponding query in the DBMS. To enable fast processing of multiple visualization recommendations simultaneously, SeeDB employs a number of optimization heuristics that are related multi-query optimization, such as merging queries with similar group-by or aggregation predicates.

4.5 Lineage/Provenance Techniques

What is it Provenance (or "lineage") refers to the recording of metadata to capture how an analysis task or operation was executed, and is actually a well-established topic of research in both the visualization [69, 142] and database communities [55, 73, 160]. Existing provenance recording techniques typically produce structured metadata, such as interaction logs or system execution logs, that users can later visualize or query to understand how certain analysis results were derived. Visualization provenance generally aims to illustrate a user's analytic reasoning process (or analytic provenance), whereas database provenance aims to record the operations and/or results from executing a data processing system (or even full workflows, known as "workflow provenance"). Database research in provenance aims to support continuous and efficient provenance recording, such that the collected metadata does not take up significant space compared to the original data, and the metadata itself can be queried quickly by the user. Database research generally employs either logical provenance (i.e., recording operations) or data provenance (i.e., recording results of operations) techniques.

Why this matters for Vis The database literature on provenance provides two opportunities that are particularly useful for the visualization community. First, one can build efficient data structures directly from provenance data to improve the performance of visualization interactions. For example, Psallidas and Wu show how provenance data can be exploited to speed up dynamic queries, specifically crossfilter [139]. More broadly, provenance data is an essential component to capturing the behavior of complex, opaque systems, where database systems are one type of opaque system, but there are others such as machine learning (especially deep learning) models [146]. These complex systems are often compute- (and data-) intensive, requiring the attention to efficiency that database research provides when recording provenance.

4.6 Indexing Techniques

What is it These techniques utilize existing data structures and/or indexing structures to enable fast and efficient processing of often spatial and/or temporal data. Data cubes, and a variant called data tiles, are examples of data structures used to facilitate fast exploration of spatiotemporal data [8, 104, 107]. R-trees are an example of a multidimensional database index used to facilitate exploration of spatial data [168, 169]. However other index types are also used to facilitate fast data lookups during interactive data exploration, such as bitmap indexes [26]. These data structures are generally pre-computed offline for a given dataset and visualization design, before exploratory visual analysis takes place, i.e., before the user ever interacts with the rendered visualization. As such, these techniques also overlap with materialized views. When done carefully, indexes can also be incrementally updated in real time based on data access patterns (e.g., data cracking [79]), which combines indexing and user modeling techniques. We discuss optimization intersections in detail in the next section.

Why this matters for Vis Since they are similar to materialized views, pre-computed data structures enable essentially constant-time lookups for the corresponding database queries. When a user's interactions are translated directly to queries, then these interactions also confer the same performance benefits. Furthermore, existing storage systems like TileDB can provide data tile specification and storage with very little overhead [131].

Though not constant-time, indexes allow for very fast retrieval of stored data as well, generally with logarithmic complexity. Furthermore, many commercial database products provide native spatial index support. For example, PostgreSQL provides native support for R-trees, which are utilized by the Kyrix visual exploration system to provide 150ms response times, on average [168, 169].

Example System Consider the Kyrix system developed by Tao et al. [168, 169]. Given a specification for a pan-zoom visualization, Kyrix pre-computes the locations and bounding boxes surrounding each mark within the visualization. Using this information, Kyrix constructs a series of spatial index structures (currently R-trees) for each zoom level, including semantic zoom levels. These R-trees are used to quickly identify and fetch which marks, and thus which tuples, are currently displayed in the viewport after each user interaction.

5 Analysis of Codes

We have organized our literature review by the high level interaction types studied by the visualization community (as summarized by Brehmer and Munzner), and the high-level classes of database optimizations identified through our coding process (see Section 4). Each cell in the table represents an application of a database optimization type to a specific interaction type. Note that multiple optimizations are often used together, and interfaces often include more than one type of interaction. As such, specific papers may appear in multiple cells. We analyze the relationships between different optimizations and interactions based on observations across the cells in our taxonomy table, and report on our findings in the remainder of this section.

5.1 "Universal" Optimization Techniques

Some optimization techniques are popular and applicable to many interaction types. Materialized views seem to improve the performance of most interaction types, and appear to be a very popular research topic. The fundamental idea behind materialized views is to save the results of previous queries, anticipating that these or similar queries will appear again in the near future. The widespread use and success of this optimization strategy for interactive visual analysis suggests there are predictable aspects to how people interact with large datasets, and we can think of view materialization as a simple form of user modeling for query prediction. One might also ask why identical or overlapping queries occur in the analysis process, and what role interaction design might play in creating this repetition. If users naturally revisit the similar visualizations (and thus similar queries) regardless of interaction design, then materialized views offer a clear advantage in terms of performance. Still, given the large body of existing literature on the creation and management of materialized views, this area may bear relatively fewer fruits in terms of new optimization opportunities.

Approximate query processing is also popular in the visualization and database communities, providing performance benefits for most interaction types. AQP methods are appealing in their wide applicability to most analytical queries: often a user's query can be answered almost as well with a small sample as with the full dataset. Furthermore, sampling techniques are already available in many commercial DBMSs,

making it straightforward to apply sampling-based optimizations within visualization systems that can connect to a DBMS. However, existing sampling methods may still incur intolerably high error in their results, or prove too restrictive in a general-purpose visualization system [32]. As a simple concrete example, we believe that giving the DBMS information about how the data will be encoded might be a valuable way to improve stratification strategies like that in BlinkDB [3].

5.2 Insights From Gaps

We offer a subjective categorization of the relative maturity of work in each subarea, based on how many references appear in each cell. We use the following color code: fewer than 10 references, 10 to 19 references, and 20 or more references. These categories seem to translate to: (1) *observations* of relevant use cases to establish tangible performance problems, (2) *algorithms* formalizing nascent optimization techniques to address established performance problems, and (3) *systems* generalizing beyond existing algorithms to provide accessible solutions for non-experts. Although coarse and necessarily limited, it provides a useful frame of reference for discussion.

5.2.1 Interaction Support Gaps

Certain interaction types are well-studied: `filter`, `aggregate`, `select`. These interactions match typical OLAP scenarios, and thus would naturally be widely supported in our survey. The next most studied interactions are `navigate` and `derive`, followed by `arrange`, and then finally `encode`, `change`, `annotate`, `record`, and `import`.

Five interaction types have empty or nearly empty rows in our table: `annotate`, `encode`, `change`, `import` and `record`. Though these interactions are useful to support in visualization systems, they appear to be irrelevant to current research efforts in terms of performance impact.

For some interactions, this outcome makes sense. `annotate` and `record` have limited throughput, because they produce one record per interaction. For example, `annotate` requires the user to annotate or augment a given visualization. Similarly, `record` requires a user to manually save visualizations. Since users tend to only interact with visual analysis tools at a rate of once every few seconds [14], data generated in this way is relatively small, and often easy to manage outside of a DBMS.

Visualization recommendation systems seem to target the intersection of user modeling optimizations and encode interactions, as they aim to anticipate the user's analysis goals, and propose relevant visualization designs to further these goals. For example, the SeeDB system [171] and CompassQL language (used in the Voyager system [181]) aim to optimize the analysis process through visualization recommendations. However, few visualization recommendation techniques currently exist, and existing systems also have their limitations (e.g., SeeDB only displays bar charts). We note that this area has seen recent growth (e.g., Draco, VizML, and Data2Vis [42, 77, 120]), but not in terms of optimization.

We were surprised by the dearth of systems that optimize for `encode` and `change`. Most papers we reviewed seem to hardcode the visualizations that users can explore, or only provide a small number of fixed visualization templates from which to construct new visualizations (e.g., only allow users to explore bar charts). Although these interaction types are known to enhance users' data exploration performance [165, 181], more research is needed to support these interactions in large datasets, or at least to evaluate the efficacy of existing optimizations.

We found that all of the papers coded with `import` describe production-level systems simultaneously supporting fast data ingest and real-time analytics on a massive (e.g., petabyte) scale. Surprisingly, most other papers seem to assume that data ingestion has already happened. **Investigating the intersection between data ingest and interactive visual analytics may also be a promising avenue for future work, given the clear industry interest observed.**

5.2.2 Optimization Application Gaps

We find few applicable examples for three optimization methods: provenance, user modeling, and MQO techniques.

We see that relatively few database optimization techniques reuse past query results, pointing to an opportunity for work on these topics for visualization use cases. However, specific optimizations target results re-use in particular ways, such as those used in Tableau [170, 177, 178]. This scarcity could come from a lack of concrete use cases for studying how results reuse, and MQO more broadly, could be applied in visualization scenarios. Recent calls for developing visualization performance benchmarks may prove useful in this case [11, 49, 182], to provide both communities with testbeds for studying under-explored optimization strategies.

The lack of concrete use cases could also be due in part to the computational complexity of baseline MQO approaches, and how difficult it is to make the baseline faster [156]. Here, **we believe that restricting MQO to visualization use cases could make MQO tractable.** Rather than supporting all of SQL, or even all of OLAP, we only have to concern ourselves with the range of possible queries supported by the visualization interface. Visualization contexts are generally limited. Visualization tools can also provide significantly more feedback about the projected utility of specific visualization designs, given the user's recent interaction history, or even the user's personality and preferences [128, 192]. Furthermore, with the popularity of visualization recommendation features (see Section 5.2.1), optimizing the computation of user requests and (possibly many) system predictions in parallel becomes attractive.

Surprisingly, we find few projects that leverage data provenance and analytic provenance for optimization purposes. Some papers leverage provenance for user modeling [21] and are in some ways similar to results reuse [139], but overall this area appears to be under-explored. Both the visualization [142] and database communities [55, 73, 160] have studied provenance topics extensively, but primarily from the perspective of illustrating this data for users (analytic provenance in visualization) or collecting it efficiently as users manipulate the data (data provenance in databases). Further, user modeling often requires access to rich provenance data from which to train the models, leading to analogous gaps in user modeling research. We believe that this intersection represents a valuable opportunity for collaboration between the two communities to develop optimization techniques that merge both analytic and data provenance.

We believe part of the issue may also stem from how in practice, only a few systems collect and manage large amounts of provenance data from multiple users over a long period of time, a notable example being VisTrails [23]. Furthermore, it is still unclear what the benefits really are (performance or otherwise) to storing a significant amount of provenance data over one or more user sessions. Presumably, if a user has to choose between storing data that will help them do their job, or storing provenance data of unknown value, then the user will likely choose to keep the former and delete the latter. As such, there is a value proposition that needs to be articulated and demonstrated to users to gain traction in this research space.

5.3 Optimization Intersections

Our coding strategy also uncovers interesting results at the intersections of optimization methods. Here we highlight existing research thrusts, and several promising future directions based on these intersections.

For example, papers that tend to have both materialized views and approximate query processing codes generally employ a pre-computed sampling strategy, where samples are constructed offline, before the user performs any data exploration [3]. These results suggest that new intersections could be developed by combining our codes, which would represent cross-cutting methods that span multiple columns or multiple rows. For example, query steering methods [43, 179] require user input to direct how approximate query processing strategies are applied, but these techniques could be combined with user modeling to predict or recommend future user interactions.

We find that user modeling is frequently combined with other optimization methods (e.g., aggregation [8, 88, 105, 164] and materialized views [152]), and is generally used to direct how these other optimizations are applied. Technically, user models are often trained on user logs, so basic provenance recording is utilized. However, to our knowledge, user modeling has not yet been combined with provenance-focused optimizations, nor indexing-based or multi-query optimization

methods. These missing combinations in our taxonomy could prove to be interesting research directions for the future.

Given that indexes are traditionally used to retrieve records, and not necessarily to derive new ones, we find that indexing-based optimizations are often paired with other optimization methods, such as materialized views [44, 101, 112, 139, 168, 169] and aggregation [22, 101, 112]. To further reduce system response times, indexing is also frequently paired with approximate query processing [34, 39, 44, 47]. In general, it seems that indexing can be combined with new optimization techniques to enhance their effectiveness.

Furthermore, it would be interesting to explore whether existing optimization methods can be applied more effectively to under-supported interaction types. For example, how might multi-query optimization techniques be designed to better support `arrange`, `encode`, and `change` interactions? As another example, could user modeling methods be extended to anticipate user analysis intent, and thus recommend future annotations (`annotate`) and queries (`derive`)?

5.4 Inspiration from Adjacent Projects

There are several papers not included directly in our literature review which were relevant in terms of motivating future research directions for optimization work. In motivating their TPFlow system, Liu et al. point out that users need "sufficient guidance" for where to search within a dataset for insights, such as patterns or anomalies [106]. Reducing the size of the data and making specific interactions fast are on their own insufficient to ensure that the user ultimately finds what they need from the data. The user still has to apply the right interactions in the right places to extract meaningful results. Similarly, Ming et al. argue that the limited screen real estate for displaying data distributions and insights should also be considered in the optimization process [67]. Thus, **it might prove effective to incorporate awareness of the user's analysis goals and environmental constraints in the process.** For example, DBMS optimizations have started to shift towards aggregating and sampling in a way that preserves salient features rather than simply making the data smaller [86, 143]. Recent work in visualization recommendation follows this approach, as does user modeling in general. Still, these ideas could be applied to all of the optimization categories we observed in our review.

6 Discussion

In this section, we contextualize and interpret our findings through our own experiences working at this research intersection, and suggest major takeaways and promising research directions for the database, visualization, and human computer interaction communities.

6.1 Distributed and Parallel Computing

Relying on parallel and distributed processing for performance reasons is a classic technique in databases and elsewhere. Although we noted a number of papers in our literature search which use such strategies (we report them in our bibliography under the `parallel` tag), we do not include a specific column for "distributed and parallel processing" in our taxonomy. We do this for the main reason that distributed and parallel techniques exist that complement each of the technique columns themselves (i.e. parallel and distributed index building, view materialization, etc). Nevertheless, we wish to highlight one specific approach in this area, that of studying the effects of *asynchrony*. It is well known that asynchronous distribution offers larger performance benefits at the cost of losing some behavioral guarantees. There is nascent work (notably by Bendre et al. and Wu et al. [17, 183]) in examining the consequences of asynchronous distribution for interactive analysis applications, but much more work is needed in the area.

6.2 Systems: Advancing Optimization

We identified six optimization types that are currently used to improve visualization system performance: materialized views, approximate query processing (AQP), user modeling, multi-query optimization (MQO), provenance, and indexing. Certain techniques, such as materialized views and AQP, appear to be very popular and universally applicable to visualization system design. Others like user modeling, MQO and provenance techniques seem to be under-explored.

Though many prototypes and techniques have been developed to apply these optimization strategies, there appears to be no consensus or standardized APIs to move them from the *algorithms* to the *systems* stage of maturity, for use by a broader audience. Though some DBMSs do provide limited access to some of this functionality, DBMS optimizations are far from standardized across product offerings [133]. Therefore, the classic advice of "just use a database" falls flat in this respect. Ongoing work in "embeddable" DBMSs (e.g., SQLite, Pandas and DuckDB [141]) have made it easier to use DBMSs directly within live programming environments, however, users are still at the mercy of what optimizations are available directly in the DBMS itself.

To make popular optimization techniques truly foundational and universally applicable, we argue that they should be shared as a standard set of packages or modular components that can easily be adopted by other communities. Basically, optimizations should be treated as first-class components within visualization systems, and implemented using a uniform structure so that current and future systems can be optimized (and evaluated) in a systematic way *outside of the DBMS*. Recent work in AQP follows this approach (e.g., VerdictDB [133]), which may provide a template for disseminating other techniques.

6.3 Algorithms: Enhancing Interaction

Though most interaction types in the literature seem to be supported by these optimization types, a few stand out. On the one hand, optimization techniques may be overkill for some underrepresented interaction types, such as `annotate` and `record`. On the other hand, some interactions are very demanding from a performance perspective, and yet are still under-supported, such as `encode` and `change` interactions. The `encode` and `change` interactions in particular appear to be particularly difficult to support at scale. Here, we suggest how the visualization community can engage other areas of computer science by clarifying the core research problems for interaction, and generally moving interaction research towards the *algorithms* stage of development.

One classic methodology in databases is to define a problem in terms of a formal (and often mathematical) theory, then design new systems and optimizations based on this formal structure. The influence of the relational model is an obvious example of the benefits of this approach, which has been developed into countless relational DBMSs, including the widely used PostgreSQL and SQL Server systems [137, 163]. The nature of visualization and human-computer interaction research makes it difficult to represent interactions with real people as formal mathematical theories. However, it is difficult for us to articulate the structure and significance of new visualization problems to other research communities without clear terminology or formalisms to share. These ideas are not necessarily new, and to some extent are captured in existing frameworks for visualization research [20, 123]. Our goal is to highlight how a more formal approach to visualization research problems not only benefits our community, but also clarifies our contributions and potential collaboration opportunities with other areas.

There are clear examples that illustrate the benefits of bridging this gap for the visualization community. For example, the formal theory behind visualization encodings has provided a useful platform for database optimization strategies, in particular automated visualization recommendations, which are a form of user modeling. In another example, systems like Polaris and DEVise proposed formal mappings from visualization designs to database queries [108, 165], enabling them to automatically offload data processing to a DBMS, as well as clarify how existing database optimization strategies (e.g., indexing, data cubes) could be exploited to improve visualization performance.

However, more work remains to be done. For example, though we have clear formalisms for static visualization designs, we still lack precise formalisms for interactions, making it difficult to clarify the differences and challenges in supporting specific interactions. New visualization languages that allow interaction specification, such as Vega and Vega-Lite [153, 155], may be useful starting points in this direction. Visualization research is often—for good reasons—presented in terms of specific applications, but a dearth of algorithms and pseudocode makes it hard to develop general, reusable approaches for the fundamental problems that the research aims to solve.

6.4 Observations: Addressing the Gaps

Though several optimization techniques are ready to move to the *systems* stage of development, we still see a surprising number of techniques that require more data collection in the *observations* stage. Specifically, more research is needed to clarify how and where under-explored optimization strategies are applicable to visualization contexts. In this work, we highlight specific gaps in the literature, such as MQO, provenance, and user modeling. For example, it could be possible that provenance optimization techniques are widely applicable to all interaction types, however we currently have insufficient empirical results to evaluate these theories. We encourage the database community to revisit these optimization areas in future work, but from the perspective of supporting interactive and visual analytics. Closer collaboration between the two communities could prove beneficial particularly for investigating provenance optimizations, since there exists a large body of work both in visualization and databases to build upon.

However in the case of MQO, we may lack publicly available use-cases or a large body of work to support new directions. A valuable first step may be to develop a benchmark or evaluation framework that showcases the challenges that MQO aims to address. Cross-community benchmarks could serve as explicit contracts between communities, a vetting of key use-cases and performance metrics to shape the development of future interactive analytics systems. Recent work makes inroads in this direction [12], but broader standards remain necessary.

6.5 Recommendations

Based on our findings and observations, we make three high-level recommendations for future research at this intersection:

R1: *Make "universal" optimization strategies standard packages or modular components that can easily be imported into other systems.*

R2: *Develop more precise formalisms and definitions for fundamental components of visualization research (e.g., interactions); and incorporate algorithms, frameworks, and pseudocode as core contributions to new visualization research projects.*

R3: *Develop an ecosystem of cross-community benchmarks to better capture the characteristics of interactive analytics, and foster collaboration around under-explored optimization areas.*

6.6 Limitations & Future Work

With this paper, we seek to highlight the breadth and depth of data management techniques that have already been applied in interactive analysis scenarios, in hopes of encouraging the visualization (and database) community to more consistently support and adopt these methods. By focusing on what has happened in the past, however, we necessarily fail to capture many current areas that might become similarly important in the future. We have already discussed the issue of asynchronous computation. Other areas we have not covered include hardware-sensitive systems design [121, 194], and compilation techniques. We hope that through more cross-area collaborations between the visualization and database communities, these new topics can be more readily adapted for interactive analysis scenarios.

Our *task analysis* is similarly backwards-looking in a visualization context. Some interactions of interest in databases are currently not well represented in visualization literature. For example, search interactions matter for information retrieval. Recent DB work attempts to formalize the relationship between search and DB systems [111], but because this style of work was not reflected in our chosen taxonomies, our analysis fails to account for it.

Finally, we also note that *system performance* alone does not necessarily improve the overall interactive analysis experience, as *human performance* must also be considered. For example, the user's ability to effectively interact with a system (independent of throughput and latency constraints) can still drive overall performance. In addition to our work, we believe there is also room for a contemporary characterization of database techniques to make visual human-data interfaces work *more effectively*, rather than merely faster.

ACKNOWLEDGMENTS

This work was partially supported by the NSF awards IIS-1815238 and IIS-1850115.

REFERENCES

[1] S. Acharya, P. B. Gibbons, and V. Poosala. Congressional samples for approximate answering of group-by queries. In *SIGMOD*. ACM, 2000.

[2] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. The Aqua Approximate Query Answering System. In *SIGMOD*. ACM, 1999.

[3] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In *EuroSys*. ACM, 2013.

[4] D. Alabi and E. Wu. PFunk-H: Approximate Query Processing Using Perceptual Models. In *HILDA*. ACM, 2016.

[5] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *InfoVis*. IEEE, 2005.

[6] B. Babcock, S. Chaudhuri, and G. Das. Dynamic Sample Selection for Approximate Query Processing. In *SIGMOD*. ACM, 2003.

[7] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale. *A Review of Temporal Data Visualizations Based on Space-Time Cube Operations*. The Eurographics Association, 2014.

[8] L. Battle, R. Chang, and M. Stonebraker. Dynamic Prefetching of Data Tiles for Interactive Visualization. In *SIGMOD*. ACM, 2016.

[9] L. Battle, R. J. Crouser, A. Nakeshimana, A. Montoly, R. Chang, and M. Stonebraker. The role of latency and task complexity in predicting visual search behavior. *TVCG*, 26(1), 2019.

[10] L. Battle et al. Position statement: The case for a visualization performance benchmark. In *Proceedings of the IEEE Workshop on Data Systems for Interactive Analysis*, Oct 2017.

[11] L. Battle et al. Evaluating visual data analysis systems: A discussion report. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA'18. ACM, 2018.

[12] L. Battle et al. Database benchmarking for supporting real-time interactive querying of large data. In *SIGMOD*. ACM, 2020.

[13] L. Battle, D. Fisher, R. DeLine, M. Barnett, B. Chandramouli, and J. Goldstein. Making Sense of Temporal Queries with Interactive Visualization. In *CHI*. ACM, 2016.

[14] L. Battle and J. Heer. Characterizing exploratory visual analysis: A literature review and evaluation of analytic provenance in tableau. In *Computer Graphics Forum*, volume 38. Wiley Online Library, 2019.

[15] L. M. Battle. *Behavior-driven optimization techniques for scalable data exploration*. PhD thesis, Massachusetts Institute of Technology, 2017.

[16] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *SIGMOD*. ACM, 1990.

[17] M. Bendre, T. Wattanawaroon, K. Mack, K. Chang, and A. Parameswaran. Anti-freeze for large and complex spreadsheets: Asynchronous formula computation. In *SIGMOD*. ACM, 2019.

[18] V. Benzaken, J.-D. Fekete, P.-L. Hémery, W. Khemiri, and I. Manolescu. EdiFlow: Data-intensive interactive workflows for visual analytics. In *ICDE*, 2011.

[19] J. Blaas, C. Botha, and F. Post. Extensions of parallel coordinates for interactive exploration of large multi-timepoint data sets. *IEEE TVCG*, 14(6), 2008.

[20] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *TVCG*, 19(12), 2013.

[21] E. T. Brown et al. Finding waldo: Learning about users from their interactions. *TVCG*, 20(12), 2014.

[22] F. Buccafurri, F. Furfaro, D. Sacca, and C. Sirangelo. A quad-tree based multiresolution approach for two-dimensional summary data. In *SSDBM*. IEEE, 2003.

[23] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. VisTrails: visualization meets data management. In *SIGMOD*. ACM, 2006.

[24] M. J. Carey and D. Kossmann. On saying "enough already!" in sql. In *SIGMOD*. ACM, 1997.

[25] R. Castro Fernandez, W. Culhane, P. Watcharapichat, M. Weidlich, V. Lopez Morales, and P. Pietzuch. Meta-Dataflows: Efficient Exploratory Dataflow Jobs. In *SIGMOD*. ACM, 2018.

[26] S. Chambi, D. Lemire, O. Kaser, and R. Godin. Better bitmap performance with Roaring bitmaps. *Software: Practice and Experience*, 46(5), 2016.

[27] C. Y. Chan and Y. E. Ioannidis. Hierarchical cubes for range-sum queries. In *VLDB*, 1999.

[28] S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining interactivity while exploring massive time series. In *VAST*. IEEE, 2008.

[29] S. Chaudhuri, G. Das, and V. Narasayya. A Robust, Optimization-based Approach for Approximate Answering of Aggregate Queries. In *SIGMOD*. ACM, 2001.

[30] S. Chaudhuri, G. Das, and V. Narasayya. Optimized Stratified Sampling for Approximate Query Processing. *ACM Trans. Database Syst.*, 32(2), 2007.

[31] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Rec.*, 26(1), 1997.

[32] S. Chaudhuri, B. Ding, and S. Kandula. Approximate query processing: No silver bullet. In *SIGMOD*, SIGMOD '17, pages 511–519, New York, NY, USA, 2017. ACM.

[33] T. Chen, H. Yin, H. Chen, R. Yan, Q. V. H. Nguyen, and X. Li. AIR: Attentional Intention-Aware Recommender Systems. In *ICDE*, 2019.

[34] X. Chen, T. Ge, J. Zhang, B. Chen, C.-W. Fu, O. Deussen, and Y. Wang. A Recursive Subdivision Technique for Sampling Multi-class Scatterplots. *IEEE TVCG*, 26(1), 2020.

[35] S.-J. Chun, C.-W. Chung, J.-H. Lee, and S.-L. Lee. Dynamic update cube for range-sum queries. In *VLDB*, 2001.

[36] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. The Case for Interactive Data Exploration Accelerators (IDEAs). In *HILDA*. ACM, 2016.

[37] A. Cuzzocrea, D. Saccà, and P. Serafino. A Hierarchy-Driven Compression Technique for Advanced OLAP Visualization of Multidimensional Data Cubes. In A. M. Tjoa and J. Trujillo, editors, *Data Warehousing and Knowledge Discovery*. Springer Berlin Heidelberg, 2006.

[38] A. Das Sarma, H. Lee, H. Gonzalez, J. Madhavan, and A. Halevy. Efficient spatial sampling of large geographical tables. In *SIGMOD*. ACM, 2012.

[39] C. A. de Lara Pahins, N. Ferreira, and J. Comba. Real-Time Exploration of Large Spatiotemporal Datasets based on Order Statistics. *IEEE TVCG*, 2019.

[40] c. Demiralp et al. Foresight: Recommending visual insights. *VLDB*, 10(12), 2017.

[41] M. Derthick, M. Christel, A. Hauptmann, and H. Wactlar. Constant density displays using diversity sampling. In *InfoVis*, 2003.

[42] V. Dibia and C. Demiralp. Data2Vis: Automatic Generation of Data Visualizations Using Sequence-to-Sequence Recurrent Neural Networks. *CG&A*, 39(5), 2019.

[43] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: an automatic query steering framework for interactive data exploration. In *SIGMOD*. ACM, 2014.

[44] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang. Sample + Seek: Approximating Aggregates with Distribution Precision Guarantee. In *SIGMOD*. ACM, 2016.

[45] H. Doraiswamy, H. T. Vo, C. T. Silva, and J. Freire. A GPU-based index to support interactive spatio-temporal queries over historical data. In *ICDE*, 2016.

[46] P. Doshi, E. Rundensteiner, and M. Ward. Prefetching for visual data exploration. In *Eighth International Conference on Database Systems for Advanced Applications, 2003. (DASFAA 2003). Proceedings.*, 2003.

[47] K. Dursun, C. Binnig, U. Cetintemel, and T. Kraska. Revisiting Reuse in Main Memory Database Systems. In *SIGMOD*. ACM, 2017.

[48] R. Ebenstein, N. Kamat, and A. Nandi. FluxQuery: An Execution Framework for Highly Interactive Query Workloads. In *SIGMOD*. ACM, 2016.

[49] P. Eichmann, E. Zgraggen, Z. Zhao, C. Binnig, and T. Kraska. Towards a benchmark for interactive data exploration. *IEEE Data Eng. Bull.*, 39(4), 2016.

[50] A. Eldawy, M. F. Mokbel, S. Alharthi, A. Alzaidy, K. Tarek, and S. Ghani. SHAHED: A MapReduce-based system for querying and visualizing spatio-temporal satellite data. In *ICDE*, 2015.

[51] C. Fan and H. Hauser. Fast and Accurate CNN-based Brushing in Scatterplots. *CGF*, 37(3), 2018.

[52] Y. Feng, D. Agrawal, A. El Abbadi, and A. Metwally. Range cube: Efficient cube computation by exploiting data correlation, 2004.

[53] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva. Visual Exploration of Big Spatio-Temporal Urban Data: A Study of New York City Taxi Trips. *IEEE TVCG*, 19(12), 2013.

[54] D. Fisher, I. Popov, S. Drucker, and m. schraefel. Trust Me, I'm Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster. In *CHI*. ACM, 2012.

[55] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3), 2008.

[56] A. Galakatos, A. Crotty, E. Zgraggen, C. Binnig, and T. Kraska. Revisiting Reuse for Approximate Query Processing. *PVLDB*, 10(10), 2017.

[57] E. Gan, J. Ding, K. S. Tai, V. Sharan, and P. Bailis. Moment-based Quantile Sketches for Efficient High Cardinality Aggregation Queries. *PVLDB*, 11(11), 2018.

[58] S. Geffner, D. Agrawal, A. E. Abbadi, and T. Smith. Relative prefix sums: an efficient approach for querying dynamic OLAP data cubes. In *ICDE*, 1999.

[59] S. Ghosh, A. Eldawy, and S. Jais. AID: An Adaptive Image Data Index for Interactive Multilevel Visualization. In *ICDE*, 2019.

[60] G. Giannikis, G. Alonso, and D. Kossmann. SharedDB: killing one thousand queries with one stone. *VLDB*, 5(6), 2012.

[61] G. Giannikis, D. Makreshanski, G. Alonso, and D. Kossmann. Shared Workload Optimization. *PVLDB*, 7(6), 2014.

[62] P. Godfrey, J. Gryz, and P. Lasek. Interactive Visualization of Large Data Sets. *IEEE TKDE*, 28(8), 2016.

[63] L. Grammel et al. How information visualization novices construct visualizations. *TVCG*, 16(6), 2010.

[64] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, 1(1), 1997.

[65] H. Guo et al. A case study using visualization interaction logs and insight metrics to understand how analysts arrive at insights. *TVCG*, 22(1), 2015.

[66] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4), 2001.

[67] M. Hao, U. Dayal, D. Keim, and T. Schreck. *Multi-Resolution Techniques for Visual Exploration of Large Time-Series Data*. The Eurographics Association, 2007.

[68] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing Data Cubes Efficiently. In *SIGMOD*. ACM, 1996.

[69] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *TVCG*, 14(6), 2008.

[70] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *ACM Queue*, 10(2), 2012.

[71] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas. Interactive data analysis: the Control project. *Computer*, 32(8), 1999.

[72] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online Aggregation. In *SIGMOD*. ACM, 1997.

[73] M. Herschel et al. A survey on provenance: What for? what form? what from? *The VLDB Journal*, 26(6):881–906, 2017.

[74] C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant. Range Queries in OLAP Data Cubes. In *SIGMOD*. ACM, 1997.

[75] P. Holanda, M. Raasveldt, S. Manegold, and H. Mühleisen. Progressive indexes: indexing for interactive data analysis. *VLDB*, 12(13), 2019.

[76] M. Hong, M. Riedewald, C. Koch, J. Gehrke, and A. Demers. Rule-based Multi-query Optimization. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. ACM, 2009.

[77] K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo. Vizml: A machine learning approach to visualization recommendation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019.

[78] E. Huang, L. Peng, L. D. Palma, A. Abdelkafi, A. Liu, and Y. Diao. Optimization for active learning-based interactive database exploration. *VLDB*, 12(1), 2018.

[79] S. Idreos, M. Kersten, and S. Manegold. Database Cracking. In *CIDR*, volume 7, 2007.

[80] J.-F. Im, K. Gopalakrishna, S. Subramaniam, M. Shrivastava, A. Tumbde, X. Jiang, J. Dai, S. Lee, N. Pawar, J. Li, and R. Aringunram. Pinot: Realtime OLAP for 530 Million Users. In *SIGMOD*. ACM, 2018.

[81] T. Isenberg, P. Isenberg, J. Chen, M. Sedlmair, and T. Möller. A systematic review on the practice of evaluating visualization. *TVCG*, 19(12), 2013.

[82] R. Jin, K. Vaidyanathan, G. Yang, and G. Agrawal. Communication and memory optimal parallel data cube construction. *IEEE TPDS*, 16(12),

2005.
[83] A. Jindal, K. Karanasos, S. Rao, and H. Patel. Selecting Subexpressions to Materialize at Datacenter Scale. *PVLDB*, 11(7), 2018.
[84] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. Interactive data exploration with smart drill-down. In *ICDE*, 2016.
[85] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. Faster Visual Analytics Through Pixel-perfect Aggregation. *PVLDB*, 7(13), 2014.
[86] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. M4: A Visualization-oriented Time Series Data Aggregation. *PVLDB*, 7(10), 2014.
[87] A. Kalinin, U. Cetintemel, and S. Zdonik. Interactive data exploration using semantic windows. In *SIGMOD*. ACM, 2014.
[88] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *ICDE*, 2014.
[89] N. Kamat and A. Nandi. A Session-Based Approach to Fast-But-Approximate Interactive Data Cube Exploration. *ACM TKDD*, 12(1), 2018.
[90] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer. Profiler: integrated statistical analysis and visualization for data quality assessment. In *AVI*. ACM, 2012.
[91] A. Kementsietsidis, F. Neven, D. Van de Craen, and S. Vansummeren. Scalable Multi-query Optimization for Exploratory Queries over Federated Scientific Databases. *PVLDB*, 1(1), 2008.
[92] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid Sampling for Visualizations with Ordering Guarantees. *PVLDB*, 8(5), 2015.
[93] A. Kim, L. Xu, T. Siddiqui, S. Huang, S. Madden, and A. Parameswaran. Optimally Leveraging Density and Locality for Exploratory Browsing and Sampling. In *HILDA*. ACM, 2018.
[94] A. Kim, L. Xu, T. Siddiqui, S. Huang, S. Madden, and A. Parameswaran. Optimally leveraging density and locality for exploratory browsing and sampling. In *HILDA*, 2018.
[95] R. Krueger, Q. Han, N. Ivanov, S. Mahtal, D. Thom, H. Pfister, and T. Ertl. Bird's-Eye - Large-Scale Visual Analytics of City Dynamics using Social Location Data. *CGF*, 38(3), 2019.
[96] B. C. Kwon, J. Verma, P. J. Haas, and C. Demiralp. Sampling for Scalable Visual Analytics. *IEEE CG&A*, 37(1), 2017.
[97] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. Empirical studies in information visualization: Seven scenarios. *TVCG*, 18(9), 2011.
[98] F. Li, B. Wu, K. Yi, and Z. Zhao. Wander Join: Online Aggregation via Random Walks. In *SIGMOD*. ACM, 2016.
[99] J. K. Li and K.-L. Ma. P5: Portable Progressive Parallel Processing Pipelines for Interactive Data Analysis and Visualization. *IEEE TVCG*, 26(1), 2020.
[100] K. Li and G. Li. Approximate Query Processing: What is New and Where to Go? *Data Science and Engineering*, 3(4), 2018.
[101] M. Li, F. Choudhury, Z. Bao, H. Samet, and T. Sellis. ConcaveCubes: Supporting Cluster-based Geographical Visualization in Large Data Scale. *CGF*, 37(3), 2018.
[102] X. Li, J. Han, and H. Gonzalez. High-dimensional OLAP: a minimal cubing approach. In *VLDB*, 2004.
[103] W. Liang, H. Wang, and M. E. Orlowska. Range queries in dynamic OLAP data cubes. *Data & Knowledge Engineering*, 34(1), 2000.
[104] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for Real-Time Exploration of Spatiotemporal Datasets. *IEEE TVCG*, 19(12), 2013.
[105] C. Liu, C. Wu, H. Shao, and X. Yuan. SmartCube: An Adaptive Data Management Architecture for the Real-Time Visualization of Spatiotemporal Datasets. *IEEE TVCG*, 26(1), 2020.
[106] D. Liu, P. Xu, and L. Ren. TPFlow: Progressive Partition and Multi-dimensional Pattern Extraction for Large-Scale Spatio-Temporal Data Analysis. *IEEE TVCG*, 25(1), 2019.
[107] Z. Liu, B. Jiang, and J. Heer. imMens: Real-time Visual Querying of Big Data. *CGF*, 32(3pt4), 2013.
[108] M. Livny et al. Devise: integrated querying and visual exploration of large datasets. *SIGMOD Record*, 26(2), 1997.
[109] Y. Luo, X. Qin, N. Tang, and G. Li. DeepEye: Towards Automatic Data Visualization. In *ICDE*, 2018.
[110] I. Mami and Z. Bellahsene. A Survey of View Selection Methods. *SIGMOD Rec.*, 41(1), 2012.
[111] B. McCamish, V. Ghadakchi, A. Termehchy, B. Touri, and L. Huang. The data interaction game. In *SIGMOD*, 2018.
[112] H. Mei, W. Chen, Y. Wei, Y. Hu, S. Zhou, B. Lin, Y. Zhao, and J. Xia. RSATree: Distribution-Aware Data Representation of Large-Scale Tabular Datasets for Flexible Visual Query. *IEEE TVCG*, 26(1), 2020.
[113] A. O. Mendelzon, A. A. Vaisman, and C. A. Hurtado. Maintaining Data Cubes under Dimension Updates. In *ICDE*, 1999.
[114] F. Miranda, M. Lage, H. Doraiswamy, C. Mydlarz, J. Salamon, Y. Lockerman, J. Freire, and C. T. Silva. Time Lattice: A Data Structure for the Interactive Visual Analysis of Large Time Series. *CGF*, 37(3), 2018.
[115] F. Miranda, L. Lins, J. T. Klosowski, and C. T. Silva. TopKube: A Rank-Aware Data Cube for Real-Time Exploration of Spatiotemporal Data. *IEEE TVCG*, 24(3), 2018.
[116] H. Mistry, P. Roy, S. Sudarshan, and K. Ramamritham. Materialized View Selection and Maintenance Using Multi-query Optimization. In *SIGMOD*. ACM, 2001.
[117] G. Moerkotte. Small Materialized Aggregates: A Light Weight Index Structure for Data Warehousing. In *VLDB*. Morgan Kaufmann Publishers Inc., 1998.
[118] D. Moritz, D. Fisher, B. Ding, and C. Wang. Trust, but Verify: Optimistic Visualizations of Approximate Queries for Exploring Big Data. In *CHI*. ACM, 2017.
[119] D. Moritz, B. Howe, and J. Heer. Falcon: Balancing Interactive Latency and Resolution Sensitivity for Scalable Linked Visualizations. In *CHI*. ACM.
[120] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. *IEEE TVCG*, 25(1), 2019.
[121] T. Mostak. Using gpus to accelerate data discovery and visual analytics. In *FTC*. IEEE, 2016.
[122] I. S. Mumick, D. Quass, and B. S. Mumick. Maintenance of Data Cubes and Summary Tables in a Warehouse. In *SIGMOD*. ACM, 1997.
[123] T. Munzner. A nested model for visualization design and validation. *TVCG*, 15(6), 2009.
[124] A. Nandi, C. Yu, P. Bohannon, and R. Ramakrishnan. Data Cube Materialization and Mining over MapReduce. *TKDE*, 24(10), 2012.
[125] R. Neamtu, R. Ahsan, E. Rundensteiner, and G. Sarkozy. Interactive time series exploration powered by the marriage of similarity distances. *VLDB*, 10(3), 2016.
[126] L. Orr, M. Balazinska, and D. Suciu. Probabilistic database summarization for interactive data exploration. *arXiv preprint arXiv:1703.03856*, 2017.
[127] A. Ottley, R. Garnett, and R. Wan. Follow The Clicks: Learning and Anticipating Mouse Interactions During Exploratory Data Analysis. *CGF*, 38(3), 2019.
[128] A. Ottley, H. Yang, and R. Chang. Personality as a predictor of user strategy: How locus of control affects search strategies on tree visualizations. In *CHI*. ACM, 2015.
[129] C. A. L. Pahins, S. A. Stephens, C. Scheidegger, and J. L. D. Comba. Hashedcubes: Simple, Low Memory, Real-Time Visual Exploration of Big Data. *IEEE TVCG*, 23(1), 2017.
[130] N. Pansare, V. Borkar, C. Jermaine, and T. Condie. Online Aggregation for Large MapReduce Jobs. In *PVLDB*, volume 4.
[131] S. Papadopoulos, K. Datta, S. Madden, and T. Mattson. The tiledb array data storage manager. *Proceedings of the VLDB Endowment*, 10(4), 2016.
[132] Y. Park, M. Cafarella, and B. Mozafari. Visualization-aware sampling for very large databases. In *ICDE*, 2016.
[133] Y. Park, B. Mozafari, J. Sorenson, and J. Wang. Verdictdb: Universalizing approximate query processing. In *SIGMOD*. ACM, 2018.
[134] P. Pedreira, C. Croswhite, and L. Bona. Cubrick: Indexing millions of records per second for interactive analytics. *VLDB*, 9(13), 2016.
[135] J. Peng, D. Zhang, J. Wang, and J. Pei. AQP++: Connecting Approximate Query Processing With Aggregate Precomputation for Interactive Analytics. In *SIGMOD*. ACM, 2018.
[136] H. Piringer, C. Tominski, P. Muigg, and W. Berger. A Multi-Threading Architecture to Support Interactive Visual Exploration. *IEEE TVCG*, 15(6), 2009.
[137] PostgreSQL. https://www.postgresql.org/. Last Accessed: 2020-04-27.
[138] M. Procopio, C. Scheidegger, E. Wu, and R. Chang. Selective Wander Join: Fast Progressive Visualizations for Data Joins. *Informatics*, 6(1), 2019.
[139] F. Psallidas and E. Wu. Smoke: Fine-grained Lineage at Interactive Speed. *PVLDB*, 11(6), 2018.
[140] L. Qiao, V. Raman, F. Reiss, P. J. Haas, and G. M. Lohman. Main-memory Scan Sharing for Multi-core CPUs. *PVLDB*, 1(1), 2008.
[141] M. Raasveldt and H. Mühleisen. Duckdb: an embeddable analytical

database. In *SIGMOD*. ACM, 2019.

[142] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *TVCG*, 22(1), 2015.

[143] S. Rahman, M. Aliakbarpour, H. K. Kong, E. Blais, K. Karahalios, A. Parameswaran, and R. Rubinfield. I've Seen "Enough": Incrementally Improving Visualizations to Support Rapid Decision Making. *PVLDB*, 10(11), 2017.

[144] V. Raman, B. Raman, and J. M. Hellerstein. Online Dynamic Reordering for Interactive Data Processing. In *VLDB*. Morgan Kaufmann Publishers Inc., 1999.

[145] C. Reach and C. North. Bandlimited OLAP cubes for interactive big data visualization. In *LDAV*, 2015.

[146] E. K. Rezig, L. Cao, M. Stonebraker, G. Simonini, W. Tao, S. Madden, M. Ouzzani, N. Tang, and A. K. Elmagarmid. Data civilizer 2.0: a holistic framework for data preparation and analytics. *Proceedings of the VLDB Endowment*, 12(12), 2019.

[147] M. Riedewald, D. Agrawal, A. E. Abbadi, and R. Pajarola. Space-Efficient Data Cubes for Dynamic Environments. In Y. Kambayashi, M. Mohania, and A. M. Tjoa, editors, *Data Warehousing and Knowledge Discovery*. Springer Berlin Heidelberg, 2000.

[148] M. Riedewald, D. Agrawal, and A. El Abbadi. Flexible Data Cubes for Online Aggregation. In J. Van den Bussche and V. Vianu, editors, *ICDT*. Springer Berlin Heidelberg, 2001.

[149] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe. Efficient and Extensible Algorithms for Multi Query Optimization. In *SIGMOD*. ACM, 2000.

[150] J. Saldaña. *The coding manual for qualitative researchers*. Sage, 2015.

[151] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In H.-J. Schek, G. Alonso, F. Saltor, and I. Ramos, editors, *Advances in Database Technology — EDBT'98*. Springer Berlin Heidelberg, 1998.

[152] S. Sarawagi and G. Sathe. I3: Intelligent, Interactive Investigation of OLAP Data Cubes. In *SIGMOD*. ACM, 2000.

[153] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *TVCG*, 23(1), 2016.

[154] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *TVCG*, 22(1), 2015.

[155] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE TVCG*, 22(1), 2016.

[156] T. Sellis and S. Ghosh. On the multiple-query optimization problem. *TKDE*, 1990.

[157] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+ -Tree: A Dynamic Index for Multi-Dimensional Objects. *SIGMOD*, 1987.

[158] T. K. Sellis. Multiple-query Optimization. *ACM Trans. Database Syst.*, 13(1), 1988.

[159] L. Sidirourgos, M. Kersten, and P. Boncz. SciBORQ: Scientific data management with Bounds On Runtime and Quality. In *Proceedings of the biennial Conference on Innovative Data Systems Research*.

[160] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *ACM Sigmod Record*, 34(3), 2005.

[161] M. Singh, A. Nandi, and H. V. Jagadish. Skimmer: rapid scrolling of relational query results. In *SIGMOD*. ACM, 2012.

[162] Y. Sismanis, A. Deligiannakis, N. Roussopoulos, and Y. Kotidis. Dwarf: Shrinking the PetaCube. In *SIGMOD*. ACM, 2002.

[163] Microsoft SQL Server. https://www.microsoft.com/en-us/sql-server/default.aspx, Last Accessed: 2020-04-27.

[164] N. Stefanovic, J. Han, and K. Koperski. Object-based selective materialization for efficient implementation of spatial data cubes. *IEEE TKDE*, 12(6), 2000.

[165] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *TVCG*, 8(1), 2002.

[166] C. Stolte, D. Tang, and P. Hanrahan. Multiscale visualization using data cubes. *IEEE TVCG*, 9(2), 2003.

[167] L. Sun, M. J. Franklin, S. Krishnan, and R. S. Xin. Fine-grained partitioning for aggressive data skipping. In *SIGMOD*. ACM, 2014.

[168] W. Tao, X. Liu, C. Demiralp, R. Chang, and M. Stonebraker. Kyrix: Interactive Visual Data Exploration at Scale. *arXiv:1905.04638 [cs]*, 2019.

[169] W. Tao, X. Liu, Y. Wang, L. Battle, C. Demiralp, R. Chang, and M. Stonebraker. Kyrix: Interactive Pan/Zoom Visualizations at Scale. *CGF*, 38(3), 2019.

[170] P. Terlecki, F. Xu, M. Shaw, V. Kim, and R. Wesley. On Improving User Response Times in Tableau. In *SIGMOD*. ACM, 2015.

[171] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis. SeeDB: efficient data-driven visualization recommendations to support visual analytics. *VLDB*, 8(13), 2015.

[172] J. Vitter and M. Wang. Approximate computation of multidimensional aggregates of sparse data using wavelets. *ACM SIGMOD Record*, 28(2), 1999.

[173] L. Wang, R. Christensen, F. Li, and K. Yi. Spatial online sampling and aggregation. *VLDB*, 9(3), 2015.

[174] Z. Wang, N. Ferreira, Y. Wei, A. S. Bhaskar, and C. Scheidegger. Gaussian Cubes: Real-Time Modeling for Visual Exploration of Large Multidimensional Datasets. *IEEE TVCG*, 23(1), 2017.

[175] A. Wasay, X. Wei, N. Dayan, and S. Idreos. Data Canopy: Accelerating Exploratory Statistical Analysis. In *SIGMOD*. ACM, 2017.

[176] Y. Wen, X. Zhu, S. Roy, and J. Yang. Interactive summarization and exploration of top aggregate query answers. *VLDB*, 11(13), 2018.

[177] R. Wesley, M. Eldridge, and P. T. Terlecki. An Analytic Data Engine for Visualization in Tableau. In *SIGMOD*. ACM, 2011.

[178] R. M. G. Wesley and P. Terlecki. Leveraging Compression in the Tableau Data Engine. In *SIGMOD*. ACM, 2014.

[179] M. Williams and T. Munzner. Steerable, Progressive Multidimensional Scaling. In *InfoVis*, 2004.

[180] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Towards a general-purpose query language for visualization recommendation. In *HILDA*. ACM, 2016.

[181] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. In *CHI*. ACM, 2017.

[182] E. Wu, L. Battle, and S. R. Madden. The case for data visualization management systems: vision paper. *VLDB*, 7(10), 2014.

[183] Y. Wu, L. Xu, R. Chang, J. M. Hellerstein, and E. Wu. Making sense of asynchrony in interactive data visualizations. *arXiv preprint arXiv:1806.01499*, 2018.

[184] K. Xu, S. Guo, N. Cao, D. Gotz, A. Xu, H. Qu, Z. Yao, and Y. Chen. ECGLens: Interactive Visual Exploration of Large Scale ECG Data for Arrhythmia Detection. In *CHI*. ACM, 2018.

[185] F. Yang, E. Tschetter, X. Léauté, N. Ray, G. Merlino, and D. Ganguli. Druid: a real-time analytical data store. In *SIGMOD*. ACM, 2014.

[186] E. T. Zacharatou, H. Doraiswamy, A. Ailamaki, C. T. Silva, and J. Freiref. GPU rasterization for real-time spatial aggregation over arbitrary polygons. *VLDB*, 11(3), 2017.

[187] K. Zeng, S. Agarwal, and I. Stoica. iOLAP: Managing Uncertainty for Efficient Incremental OLAP. In *SIGMOD*. ACM, 2016.

[188] E. Zgraggen, A. Galakatos, A. Crotty, J. Fekete, and T. Kraska. How Progressive Visualizations Affect Exploratory Analysis. *IEEE TVCG*, 23(8), 2017.

[189] C. Zhan, M. Su, C. Wei, X. Peng, L. Lin, S. Wang, Z. Chen, F. Li, Y. Pan, F. Zheng, and others. AnalyticDB: real-time OLAP database system at Alibaba cloud. *VLDB*, 12(12), 2019.

[190] L. Zhang, A. Stoffel, M. Behrisch, S. Mittelstadt, T. Schreck, R. Pompl, S. Weber, H. Last, and D. Keim. Visual analytics for the big data era — A comparative review of state-of-the-art commercial systems. In *VAST*, 2012.

[191] B. Zhou and Y.-J. Chiang. Key Time Steps Selection for Large-Scale Time-Varying Volume Datasets Using an Information-Theoretic Storyboard. *CGF*, 37(3), 2018.

[192] C. Ziemkiewicz, A. Ottley, R. J. Crouser, A. R. Yauilla, S. L. Su, W. Ribarsky, and R. Chang. How visualization layout relates to locus of control and other personality factors. *TVCG*, 19(7), 2012.

[193] K. Zoumpatianos, S. Idreos, and T. Palpanas. Indexing for interactive exploration of big data series. In *SIGMOD*. ACM, 2014.

[194] M. Zukowski, P. A. Boncz, N. Nes, and S. Héman. Monetdb/x100-a dbms in the cpu cache. *IEEE Data Eng. Bull.*, 28(2), 2005.

[195] M. Zukowski, S. Héman, N. Nes, and P. Boncz. Cooperative Scans: Dynamic Bandwidth Sharing in a DBMS. In *PVLDB*. VLDB Endowment, 2007.