Towards Democratizing Relational Data Visualization

Nan Tang
QCRI
ntang@hbku.edu.qa

Eugene Wu
Columbia University
ewu@cs.columbia.edu

Guoliang Li
Tsinghua University
liguoliang@tsinghua.edu.cn

ABSTRACT

The problem of data visualization is to transform data into a visual context such that people can easily understand the significance of data. Nowadays, data visualization becomes especially important, because it is the *de facto* standard for modern business intelligence and successful data science. This tutorial will cover three specific topics: ***visualization languages*** define how the users can interact with various visualization systems; ***efficient data visualization*** processes the data and produces visualizations based on well-specified user queries; ***smart data visualization*** recommends data visualizations based on underspecified user queries. In this tutorial, we will go logically through these prior art, paying particular attentions on problems that may attract the interest from the database community.

1 INTRODUCTION

Data visualization, which transforms abstract data into visual representations (for example, length, position, shape, color, and so on), is a powerful means to present compelling stories of data to humans who are more visually oriented. Nowadays, all organizations have more data than ever at their disposal. Consequently, more and more organizations use data and advanced analytics to inform strategic and operational decisions. Data visualization is a natural means to both provide an overview of massive datasets, and to help users interpret the results of data analytics. The success of leading vendors in data visualization, such as Tableau [9], Qlik [7] and Power BI [5], has revolutionized the way that even non-technical people can understand and take action through data.

The considerable interest and efforts from industry and academia have gone a long way, however, the journey of *democratizing relational data visualization* – such that anyone can easily generate various visualizations to understand, analyses, and present massive volumes of data – is still in its early stages. There continues to be significant and important challenges. The purpose of this tutorial is to provide an overview of where we are from the data management perspective.

SIGMOD'19, June 30–July 05, 2019, Amsterdam, Netherlands
© 2019 Association for Computing Machinery.
https://doi.org/10.1145/3183713.3193545

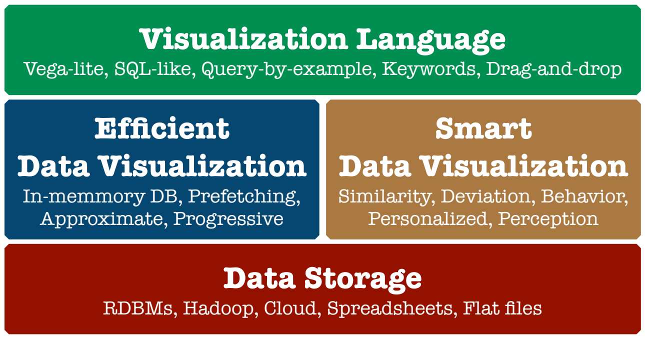
Visualization Language
Vega-lite, SQL-like, Query-by-example, Keywords, Drag-and-drop
Efficient Data Visualization
In-memmory DB, Prefetching, Approximate, Progressive
Smart Data Visualization
Similarity, Deviation, Behavior, Personalized, Perception
Data Storage
RDBMs, Hadoop, Cloud, Spreadsheets, Flat files

Figure 1: The Stack of Data Visualization

Problem Overview and Tutorial Scope. Let us provide a small amount of formalism. *Given a relational database* $\mathbf{D}$*, a user specification* $\mathbf{S}$*, a* ***data visualization*** *can be thought of as a function* $\mathbf{F}(\cdot,\cdot)$ *such that* $\mathbf{F}(\mathbf{D},\mathbf{S})$ *computes a set* $\mathbf{V}$ *of visualizations.*

Based on the above definition, Figure 1 presents a simple architectural stack for relational data visualization.

Data storage describes the data storage layer containing the relational data $\mathbf{D}$. In practice, it may be stored in a DBMS, cloud storage, or flat files. It may be very large: it may contain thousands of tables (or databases), tables may be wide (*e.g.,* several hundred attributes), and the volumes may be large (*e.g.,* petabytes). Most commercial visualization systems can access and visualize across multiple sources, such as Tableau [9], Qlik [7], Amazon QuickSight [1]. In contrast, most research prototypes mainly read data from DBMSs, such as SeeDB [77], DeepEye [61], Polaris [75], zenvisage [73], and Voyager [81].

We mention storage for completeness, however leave the technical details out of this tutorial, because it is the focus of entire subfields of the data management community. We will instead focus on the three visualization-specific components below:

Visualization languages are the mechanism that users express their visualization goals. A language specification S must be precise enough that the visualization system can process it efficiently, yet easy to use for developers or end-users. Language specification vary from procedural to declarative, and can be expressed textually or interactively (e.g., drag-and-drop operations).

Efficient data visualization describes the procedure to quickly generate visualizations **V** when the specification **S** is well-specified, *i.e.,* there is no ambiguity of what **F**(**D**, **S**) will produce. A primary goal is responsiveness and scalability (*i.e.,* **F**(**D**, **S**)) can be evaluated in real time); to achieve this goal, systems may relax the result semantics to return approximate and progressive visualizations if **F**(**D**, **S**)) cannot be evaluated in real time.

Smart data visualization is when the user underspecifies S. The system must first "smartly" complete the specification in a way that anticipates the user's intentions, before it can render a visualization. This is analogous to keyword search, in that the search terms are underspecified. Thus different search engines (e.g., Google, Bing) will output different results. Smart data visualization could be fully automatic, reference visualization based, user behavior based, personalized, and so forth.

Challenges and Open Problems. The last part of this tutorial is dedicated to the challenges and open problems, which include big data challenges due to massive datasets that still stymie the goal of real time interactions, the effects of (dirty) data whose resulting visualizations may mislead the users with false trends due to data errors, deep data visualization that leverages deep learning technologies towards smarter data visualization, a similarly large data visualization benchmark for smart data visualization recommendations. and other related problems.

2 TARGET AUDIENCE AND LENGTH

The primary audience is researchers, practitioners and students that are interested in data visualization, or using data visualization to solve problems. The tutorial will be self-contained, and we will include a broad introduction and motivating examples for non-specialists to follow.

Moreover, this tutorial should also be of interest to the data mining and machine learning communities, due both to the importance of finding compelling stories through data visualization for various data analytical tasks, and to its connections with smart data visualizations.

The intended length of this tutorial is 1.5 hours.

3 TUTORIAL OUTLINE

We shall organize the tutorial as follows.

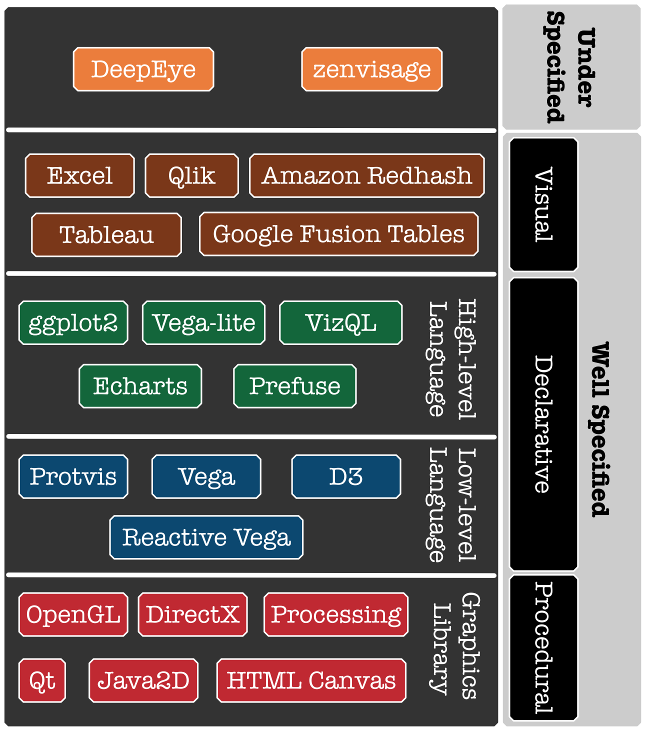

Figure 2: Data Visualization Languages

3.1 Introduction and Overview

We will start with a brief overview of this tutorial, to give the audience a clear outline and talk goals. We will then present motivating examples from emerging applications to illustrate the importance of data visualization in multiple domains and tasks.

3.2 Data Visualization Languages

In this tutorial, we use the term "data visualization languages" to broadly mean any method that the users can specify *what* visualizations do they want to data visualization systems.

On one side, users can be either experts or non-experts, and may come from different backgrounds, such as mathematician or engineer. On the other side, data visualization systems are implemented based on different design purposes, personal expertise, or any other reason. Consequently, there are many data visualization languages. Figure 2 gives an overview about our classification of visualization languages.

Well-specified Languages. This class of languages provide unambiguous semantics to a given query S. We define three classes of well-specified languages.

(1) *Procedural language* requires the specification of a series of well-structured steps and procedures to compose a visualization. They are typically *Graphics Libraries*, which specify

the most basic graphical primitives (*e.g.,* vector in vector graphic, pixel in bitmap) in graphics, such as OpenGL, DirectX, Qt, Java2D, HTML Canvas, Raphaël [8], Processing [6], Piccolo [17] and others.

(2) *Declarative language* requires the specification $\mathbf{S}$ of only *what* the users want; *how* to execute them is engine dependent (*i.e.,* the implementation of function $\mathbf{F}(\mathbf{D}, \mathbf{S})$). We further categorize them into two classes.

- *Low-level languages* [10, 19, 49, 69, 79] abstract graphical elements as visualization primitives. The visualization primitives include marks (*e.g.,* bar in histogram, line in line chart), axes, legends, scales (the mapping of data to visual attributes, usually as a function), transformations (*e.g.,* grouping and binning), signals (used in interaction), and so on. We will mainly talk about D3 [49, 79], Vega [10] and Reactive Vega [69].
- *High-level languages* [2, 3, 32, 33, 42, 68, 79, 80, 82, 86] further abstract the details of low-level visualization construction. They provide concise specification interfaces that are easier for new users to learn and use. Users omit low-level details, which a compiler fills in with sensible defaults. In this context, we will focus on key ideas from ggplot2 [79] and Vega-Lite [68], and discuss their connections with relational languages such as SQL.

(3) *Visual languages* follow the "direct manipulation principle" [71], that lets end-users interact with a visual interface to specify visual operations and specifications through a mouse or touch screen. For instance, Polaris [75] lets users directly drag attributes from a list onto the X or Y axes to specify aggregation-based analyses. Interactions may also specify transformations, chart types, predicates, and more.

Visual languages are widely, and increasingly, used in commercial visualization systems such as Microsoft Excel, Apple Numbers, Amazon Redash, Qlik, Tableau [9] (evolved from Polaris [75]), Spotfire [12], Google Spreadsheets, Google Fusion Tables [29] and more.

Underspecified Languages. Generally speaking, underspecific languages contain "gaps" in the specification $\mathbf{S}$, and it is the task of the visualization system to interpret the underspecified input (*i.e.,* the implementation of the function $\mathbf{F}(\mathbf{D}, \mathbf{S})$), in a variety of ways. Users may leave "hints" in the gaps so the system can better interpret.

The first type of hint is "reference-based", where the users provide a reference visualization as a seed. zenvisage [72, 73] supports queries which return similar or dissimilar visualizations (*e.g.,* similar trends in line charts) with a user provided reference visualization. Similarly, Draco [51], Voyager2 [81] and others take as input incomplete declarative visualization specifications as reference.

The second type of hint is "keyword-based", similar to keyword search. Systems such as VizDeck [55] and DeepEye [46][1] take as input keywords and return recommended visualizations. For example, the user of the latter system may input "*show me line charts about electricity*", and the system will recommend line charts which also contain the column "electricity".

3.3 Efficient Data Visualization

When the visualization specification is well-defined, a key system goal is to generate the visualization quickly. We will present four important classes of techniques.

(1) Exact data visualization. The class describes systems that compute visualizations exactly. We further refine these based on their system design:

- *Apply existing systems and technique.* Many visualization systems (*e.g.,* DeepEye [45, 61], Polaris [75], SeeDB [77, 78], Vizdom [21], M4 [37]) perform computation via issuing SQL queries to a general DBMS. Others [27, 48, 56, 64] use hardware (such as multicore and GPU), parallel processing [44, 56, 77, 78], or existing types of data structures to accelerate visualization.
- *Modify DBMS designs.* Industry and academia have modified DBMS functionality or system designs to optimize them towards data visualization. These include industry systems such as Hyper [4, 53, 54] which is an efficient main-memory and hybrid OLTP and OLAP DBMS that is now customized for Tableau's data engine. In academia, systems such as the Data Visualization Management System [58–60, 82, 87] explore how relational query languages can be used to express interactive visualizations [87], and how relational DBMS designs can be adapted and extended – for instance with fast lineage support [58–60] – to speed up interactive visualization.
- *Predictive data visualization.* Users interact with visualizations throughout their exploration and analysis process. These interactions are typically informed from current and previous views—by changing the parameters of current visualization, adding a predicate, or zooming to see details or overviews. Thus many visualization systems [16, 18, 20, 25, 35, 38, 44, 76] seek to predict and speculatively execute/prefetch future visualizations that the user may request.

(2) Approximate data visualization. In some cases, it may be too difficult to compute exact visualizations quickly enough. In these cases, approximation is a pragmatic and effective mechanism to trade-off responsiveness and accuracy,

[1]http://deepeye.tech

and many systems [11, 13, 23, 24, 34, 41, 50, 57, 63] speed up data processing by leveraging approximate query processing (AQP) techniques. To inform how error bounds may be set, there has been recent work to quantify perceptual inaccuracies [65, 83], or *perceptual function*, in ways that may be employed by such approximation approaches.

(3) Progressive data visualization. A related, but distinct concept, is to quickly provide an overview (or approximation) of the visualization, and then gradually refine the details over time [18, 23, 26, 35, 43, 50, 63, 74]. Users have the flexibility to choose to either wait for more interesting details, or be sufficiently satisfied to make a decision (e.g., perform another interaction). It is important to understand how progressive results affect the user's exploration process [88].

(4) Data reduction. When rendering visualizations, many systems map each data point to a visual element, which may result in a cluttered visualization that overloads the user. Data reduction methods help address this issue by summarizing the dataset to help users better identify patterns. We will review common data reduction techniques – including filtering and sampling [23, 41, 50, 67], aggregation (clustering) [31], and model fitting [28] – and when each is effective for particular tasks.

3.4 Smart Data Visualization

Precisely specifying S is hard, even for experts, especially in the common situation that the users may not even know what they precisely want. Not surprisingly, several systems [46, 62, 72, 73] allow users to provide an underspecified S, and smartly computes visualizations **V**, which is also referred to as *data visualization recommendation*. The practical need for such systems is that data visualization is typically used for data exploration – users try to find the compelling but unknown stories instead of just depicting the stories.

Apparently, the hardest part of smart data visualization is to quantify the "goodness" of visualizations **V** *w.r.t.* **D** and **S**, *i.e.,* to "guess" what the users want. Intuitively, the "one size does not fit all" principle applies here. Consequently, which visualizations to recommend have been approached from different angles, such as similarity-based methods [72, 73] that recommend visualizations which have the similar trends, patterns or statistical information with the reference visualization; deviation-based approaches [40, 72, 77, 78] that capture interesting visualizations as outliers; behavior-based solutions that infer users' intent by his/her present behavior [30], or a sequence of actions during interactions; personalized visualizations that recommend visualizations by leveraging historical data [55]; and perception-based methods that model the perceptual effectiveness by predefined rules [47, 61, 70, 81] or using machine learning models [22, 45, 51].

Furthermore, even if **S** is well-specified, recommending more interesting visualizations, similar to or different from **V**, to the users could still have plenty of rewarding reasons.

3.5 Conclusions and Open Problems

Although data visualization has been extensively studied, there are still many opportunities and challenges towards the goal of democratizing data visualization. The tutorial focuses on the above challenges, however we will also discuss several open problems:

Interactivity under massive data volumes. Many existing approaches towards interactivity may be reduced to processing a single visualization faster, on more data. However, there is still tremendous opportunity to explicitly model and address interactivity directly – visualization specifications triggered by interactions are highly correlated. Modeling interactions as *sessions* of similar queries [85] can enable explicit state sharing across queries.

Dirty data. Real-life data is typically dirty and visualizing dirty data may mislead users. For example, a data that is integrated from multiple sources may contain many duplicates. Carefully cleaning every dataset before visualizing them is simply too expensive in practice. Hence, visualization-driven data cleaning that focuses on the user's specific analysis is in high demand, and early approaches such as Scorpion [84], Profiler [40], and others [39, 52] push towards this goal.

Deep learning for smarter data visualization. Deep learning based techniques have revolutionized many domain, such as image recognition, natural language understand, automatic car, and many others. An emerging topic is the use of deep learning to better recommend data visualizations [61, 66], in an analagous way as used in modern search engines.

Data Visualization Benchmarks. Like ImageNet or the classic TPC benchmarks, it is important to develop benchmarks for performance and recommendation. The benchmarks should be faithful to the visual analysis tasks, provide reusable traces and data, and in the case of recommendation, have high coverage and quality of its labels. There is an emerging focus on developing benchmarks for performance measures [14, 15, 36]. There is potential for a similarly large data visualization benchmark for smart data visualization recommendations.

4 BIOGRAPHY

Nan Tang is a Senior Scientist at Qatar Computing Research Institute (QCRI), HBKU, Qatar. Before that he was a research fellow at the University of Edinburgh, Scotland, UK, and he also worked as a Scientific Staff Member at CWI, Amsterdam, The Netherlands. He has received the best paper award of VLDB 2010, and several papers have been invited to VLDBJ and TKDE as the best papers series. Dr Nan's main research interests are democratizing data (*i.e.,* data discovery, integration and cleaning), democratizing visualization (*i.e.,* semi-automatic data visualization, and visualization in virtual/augmented reality), and democratizing analytics (*i.e.,* making data analytics easy for non-experts).

Eugene Wu is an assistant professor at Columbia University, New York, USA. Eugene's primary research interests focus on the systems and algorithmic challenges at the intersection of databases and interactive visual interfaces, and more broadly, for the future of human-data-interaction. He has received the 2018 VLDB 10-year test of time award, best of conference citations at ICDE and VLDB, SIGMOD 2016 best demo award, and faculty awards from Google and Amazon. He is funded by NSF 1527765, NSF 1564049.

Guoliang Li is an associate professor at Tsinghua University, Beijing, China. Guoliang's main research interests include data integration and cleaning, data visualization, and human-in-the-loop data management. He got VLDB 2017 early research contribution award, TCDE 2014 early career award, CIKM 2017 best paper award. His ICDE 2018 and KDD 2018 papers have been invited to TKDE and TKDD as the best papers series.

REFERENCES

[1] Amazon quicksight: Cloud based business intelligence. https://aws.amazon.com/quicksight/.

[2] Echarts. http://echarts.baidu.com.

[3] Flare. http://flare.prefuse.org.

[4] Hyper: A hybrid oltp&olap high performance dbms. https://hyper-db.de.

[5] Power bi: Interactive data visualization bi tools. https://powerbi.microsoft.com.

[6] Processing. https://processing.org.

[7] Qlik: Data analytics for modern business intelligence. https://www.qlik.com/us.

[8] Raphaël: Javascript library. http://raphaeljs.com.

[9] Tableau. https://www.tableau.com.

[10] Vega: A visualization grammar. https://vega.github.io/vega/.

[11] 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 *ACM*, 2013.

[12] C. Ahlberg. Spotfire: an information exploration environment. In *SIGMOD Record*. ACM, 1996.

[13] D. Alabi and E. Wu. Pfunk-h: approximate query processing using perceptual models. In *HILDA*, 2016.

[14] L. Battle, M. Angelini, C. Binnig, T. Catarci, P. Eichmann, J.-D. Fekete, G. Santucci, M. Sedlmair, and W. Willett. Evaluating visual data analysis systems: A discussion report. In *HILDA*, 2018.

[15] L. Battle, R. Chang, J. Heer, and M. Stonebraker. Position statement: The case for a visualization performance benchmark. In *DSIA Workshop*, 2017.

[16] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. In *SIGMOD*, 2015.

[17] B. B. Bederson, J. Grosjean, and J. Meyer. Toolkit design for interactive structured graphics. In *IEEE TSE*, 2004.

[18] N. Bikakis, G. Papastefanatos, M. Skourla, and T. Sellis. A hierarchical aggregation framework for efficient multilevel visual exploration and analysis. In *Computer Science*, 2016.

[19] M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. In *IEEE Transactions on Visualization & Computer Graphics*, 2009.

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

[21] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. Vizdom: Interactive analytics through pen and touch. In *PVLDB*, 2015.

[22] V. Dibia and Ç. Demiralp. Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. In *arXiv*, 2018.

[23] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang. Sample + seek: Approximating aggregates with distribution precision guarantee. In *SIGMOD*, 2016.

[24] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang. Sample+ seek: Approximating aggregates with distribution precision guarantee. In *SIGMOD*, 2016.

[25] P. R. Doshi, E. A. Rundensteiner, and M. O. Ward. Prefetching for visual data exploration. In *Masters Theses*, 2003.

[26] N. Elmqvist and J. D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. In *IEEE Transactions on Visualization & Computer Graphics*, 2010.

[27] J. Fekete and C. Plaisant. Interactive information visualization of a million items. In *The Craft of Information Visualization*, 2003.

[28] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick. Online segmentation of time series based on polynomial least-squares approximations. In *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2010.

[29] H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, and J. Goldberg-Kidon. Google fusion tables: web-centered data management and collaboration. In *SIGMOD*, 2010.

[30] D. Gotz and Z. Wen. Behavior-driven visualization recommendation. In *IUI*, 2009.

[31] G. Hackenbroich, G. Hackenbroich, G. Hackenbroich, and V. Markl. Faster visual analytics through pixel-perfect aggregation. In *PVLDB*, 2014.

[32] P. Hanrahan. Vizql:a language for query, analysis and visualization. In *SIGMOD*, 2006.

[33] J. Heer, S. K. Card, and J. A. Landay. Prefuse: a toolkit for interactive information visualization. In *CHI*, 2005.

[34] 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. In *Computer*. IEEE, 1999.

[35] P. Jayachandran, K. Tunga, N. Kamat, and A. Nandi. Combining user interaction, speculative query execution and sampling in the dice system. In *PVLDB*, 2014.

[36] L. Jiang, P. Rahman, and A. Nandi. Evaluating interactive data systems: Workloads, metrics, and guidelines. In *SIGMOD*, 2018.

[37] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. M4: a visualization-oriented time series data aggregation. In *PVLDB*, 2014.

[38] A. Kalinin, U. Cetintemel, and S. Zdonik. Interactive data exploration using semantic windows. In *SIGMOD*, 2014.

[39] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Wrangler: interactive visual specification of data transformation scripts. In *CHI*, 2011.

[40] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer. Profiler: integrated statistical analysis and visualization for data quality assessment. In *AVI*, 2012.

[41] A. Kim, E. Blais, A. G. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees.

In *PVLDB*, 2015.
[42] D. Li, H. Mei, Y. Shen, S. Su, W. Zhang, J. Wang, M. Zu, and W. Chen. Echarts: A declarative framework for rapid construction of web-based visualization. In *Visual Informatics*, 2018.
[43] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. In *IEEE Transactions on Visualization & Computer Graphics*, 2013.
[44] Z. Liu, B. Jiang, and J. Heer. immens : real-time visual querying of big data. In *EuroVIS*, 2013.
[45] Y. Luo, X. Qin, N. Tang, and G. Li. Deepeye: Towards automatic data visualization. In *ICDE*, 2018.
[46] Y. Luo, X. Qin, N. Tang, G. Li, and X. Wang. Deepeye: Creating good data visualizations by keyword search. In *SIGMOD*, 2018.
[47] J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. In *TVCG*. IEEE, 2007.
[48] B. McDonnel and N. Elmqvist. Towards utilizing gpus in information visualization: A model and implementation of image-space operations. In *TVCG*. IEEE, 2009.
[49] B. Michael, O. Vadim, and H. Jeffrey. D3: Data-driven documents. In *IEEE Transactions on Visualization & Computer Graphics*, 2011.
[50] D. Moritz, D. Fisher, B. Ding, and C. Wang. Trust, but verify: Optimistic visualizations of approximate queries for exploring big data. In *CHI*, 2017.
[51] 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. In *InfoVIS*, 2018.
[52] K. Morton, H. Hajishirzi, M. Balazinska, and D. Grossman. View-driven deduplication with active learning. In *arXiv*, 2016.
[53] T. Neumann. *Efficiently compiling efficient query plans for modern hardware*. VLDB Endowment, 2011.
[54] T. Neumann and A. Kemper. Fast serializable multi-version concurrency control for main-memory database systems. In *SIGMOD*, 2015.
[55] D. B. Perry, B. Howe, A. M. Key, and C. Aragon. Vizdeck: Streamlining exploratory visual analytics of scientific data. In *iConference*, 2013.
[56] H. Piringer, C. Tominski, P. Muigg, and W. Berger. A multi-threading architecture to support interactive visual exploration. In *TVCG*, 2009.
[57] M. Procopio, C. Scheidegger, E. Wu, and R. Chang. Load-n-go: Fast approximate join visualizations that improve over time. In *DSIA*, 2017.
[58] F. Psallidas and E. Wu. Demonstration of smoke: A deep breath of data-intensive lineage applications. In *SIGMOD (demo)*, 2018.
[59] F. Psallidas and E. Wu. Provenance in interactive visualizations. In *HILDA*, 2018.
[60] F. Psallidas and E. Wu. Smoke: Fine-grained lineage at interactive speeds. In *VLDB*, 2018.
[61] X. Qin, Y. Luo, N. Tang, and G. Li. Deepeye: An automatic big data visualization framework. In *Big Data Mining & Analytics*, 2018.
[62] X. Qin, Y. Luo, N. Tang, and G. Li. Deepeye: Visualizing your data by keyword search. In *EDBT Vision*, 2018.
[63] S. Rahman, M. Aliakbarpour, H. K. Kong, E. Blais, K. Karahalios, A. Parameswaran, R. Rubinfield, S. Rahman, M. Aliakbarpour, and H. K. Kong. I've seen "enough": incrementally improving visualizations to support rapid decision making. In *PVLDB*, 2017.
[64] O. Rübel, K. Wu, H. Childs, J. Meredith, C. G. Geddes, E. Cormier-Michel, S. Ahern, G. H. Weber, P. Messmer, H. Hagen, et al. High performance multivariate visual data exploration for extremely large data. In *SC*, 2008.
[65] G. Ryan, A. Mosca, R. Chang, and E. Wu. At a glance: Approximate entropy as a measure of line chart visualization complexity. In *InfoVIS*, 2018.
[66] B. Saket, D. Moritz, H. Lin, V. Dibia, C. Demiralp, and J. Heer. Beyond heuristics: Learning visualization design. In *arXiv*, 2018.
[67] A. D. Sarma, H. Lee, H. Gonzalez, J. Madhavan, and A. Halevy. Efficient spatial sampling of large geographical tables. In *SIGMOD*, 2012.
[68] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. In *IEEE Transactions on Visualization & Computer Graphics*, 2016.
[69] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. In *IEEE Transactions on Visualization & Computer Graphics*, 2015.
[70] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. In *Information Visualization*, 2005.
[71] B. Shneiderman. Direct manipulation: A step beyond programming languages. In *IEEE Computer*, 1983.
[72] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. G. Parameswaran. Effortless data exploration with zenvisage: An expressive and interactive visual analytics system. In *PVLDB*, 2016.
[73] T. Siddiqui, J. Lee, A. Kim, E. Xue, X. Yu, S. Zou, L. Guo, C. Liu, C. Wang, K. Karahalios, and A. G. Parameswaran. Fast-forwarding to desired visualizations with zenvisage. In *CIDR*, 2017.
[74] C. D. Stolper, A. Perer, and D. Gotz. Progressive visual analytics: User-driven visual exploration of in-progress analytics. In *IEEE Transactions on Visualization & Computer Graphics*, 2014.
[75] C. Stolte and P. Hanrahan. Polaris: a system for query, analysis and visualization of multi-dimensional relational databases. In *Information Visualization*, 2000.
[76] F. Tauheed, T. Heinis, F. ShÃijrmann, H. Markram, and A. Ailamaki. Scout: Prefetching for latent feature following queries. In *PVLDB*, 2012.
[77] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: automatically generating query visualizations. In *PVLDB*, 2014.
[78] M. Vartak, S. Rahman, S. Madden, A. G. Parameswaran, and N. Polyzotis. SEEDB: efficient data-driven visualization recommendations to support visual analytics. In *PVLDB*, 2015.
[79] H. Wickham. A layered grammar of graphics. In *Journal of Computational & Graphical Statistics*, 2010.
[80] L. Wilkinson. *The grammar of graphics*. Springer, 2005.
[81] 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*, 2017.
[82] E. Wu, L. Battle, and S. R. Madden. The case for data visualization management systems. In *Cancer Letters*, 2014.
[83] E. Wu, L. Jiang, L. Xu, and A. Nandi. Graphical perception in animated bar charts. In *arXiv*, 2016.
[84] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. In *VLDB*, 2013.
[85] E. Wu and A. Nandi. Towards perception-aware interactive data visualization systems. In *DSIA Workshop*, 2015.
[86] E. Wu, F. Psallidas, Z. Miao, H. Zhang, and L. Rettig. Combining design and performance in a data visualization management system. In *CIDR*, 2017.
[87] E. Wu, F. Psallidas, Z. Miao, H. Zhang, L. Rettig, Y. Wu, and T. Sellam. Combining design and performance in a data visualization management system. In *CIDR*, 2017.
[88] E. Zgraggen, A. Galakatos, A. Crotty, J.-D. Fekete, and T. Kraska. How progressive visualizations affect exploratory analysis. In *IEEE Transactions on Visualization & Computer Graphics*. IEEE, 2017.