Partial Adaptive Indexing for Approximate Query Answering

Stavros Maroulis
ATHENA Research Center, Greece

Nikos Bikakis
Hellenic Mediterranean University &
ATHENA Research Center, Greece

Vassilis Stamatopoulos
ATHENA Research Center, Greece

George Papastefanatos
ATHENA Research Center, Greece

ABSTRACT

In data exploration, users need to analyze large data files quickly, aiming to minimize data-to-analysis time. While recent adaptive indexing approaches address this need, they are cases where demonstrate poor performance. Particularly, during the initial queries, in regions with a high density of objects, and in very large files over commodity hardware. This work introduces an approach for adaptive indexing driven by both query workload and user-defined accuracy constraints to support approximate query answering. The approach is based on partial index adaptation which reduces the costs associated with reading data files and refining indexes. We leverage a hierarchical tile-based indexing scheme and its stored metadata to provide efficient query evaluation, ensuring accuracy within user-specified bounds. Our preliminary evaluation demonstrates improvement on query evaluation time, especially during initial user exploration.

VLDB Workshop Reference Format:
Stavros Maroulis, Nikos Bikakis, Vassilis Stamatopoulos, and George Papastefanatos. Partial Adaptive Indexing for Approximate Query Answering. VLDB 2024 Workshop: 7th International Workshop on Big Data Visual Exploration and Analytics (BigVis 2024).

1 INTRODUCTION

In data exploration, users often need to *interact with and analyze* large data files without the hassle of full-fledged database configuration and lengthy data loading and indexing times. A common objective in such scenarios is to minimize the *data-to-analysis time* while ensuring efficient visual exploration and analytical operations.

In such exploration scenarios, users might not always *require exact results*. There are interactive scenarios where response time is more crucial than result accuracy [6, 9, 14, 16, 18, 20]. Several visual analytic tasks, such as class or outlier analysis in scatterplots, pair-wise comparison of spatial areas on maps, usually start with approximate aggregated insights, which can be used by the experts to quickly identify specific areas in the exploration space for further analysis. Approximate query processing (AQP) is a long-studied problem in the areas of databases and information visualization; however, there is a missing gap when AQP is coupled with the *in-situ paradigm*.

In-situ paradigm has been adopted for data exploration, enabling on-the-fly analysis of large raw data sets such as CSV or JSON files [2, 3, 11, 13]. In-situ techniques aim to bypass the overhead of fully loading and indexing data in a DBMS while offering efficient query evaluation over the raw data files. Previous works on the visual exploration of raw data files [3, 11] have focused on building adaptive indexes over the raw data objects. These methods leverage the locality-based behaviour of visual exploration, in which the user explores a specific area in the data space, gradually widening the analysis in neighboring areas. Thus, they seek to minimize index initialization time by initially creating a "crude" version of the index, e.g., around the initial area of interest, dynamically extending and adapting it based on user exploration.

However, since the initial index is only a "crude" version, *initial queries can be slower as the index adapts and refines based on user exploration*. Additionally, *for very large raw data files or regions with a high density of objects*, even an index that has sufficiently adapted can result in significant query times, thus compromising interactivity.

Considering these challenges, *our goal is to reduce response time by providing approximate results and performing "partial" index adaptation*. Partial index adaptation allows us to reduce the costs associated with reading the raw data file (i.e., I/O cost) and refining the index, e.g., recomputing metadata, reorganizing objects.

Contribution. In this paper we provide our ongoing work and preliminary results on the problem of *adaptive indexing driven by both the query workload and the accuracy constraints set by the user*. Particularly, *given an accuracy constraint we attempt to reduce the response time, by performing a minimum number of adaptations (and consequence the I/O's) such as the result's accuracy is larger than a threshold.* The proposed methods are going to be integrated to our *RawVis framework*[1] [3, 10, 11], supporting approximate query answering via partial index adaptation.

Related Work. In visual analytics, *approximate processing* techniques (a.k.a. *data reduction*), such as sampling and binning, have been used to improve efficiency and address the visual information overloading problem [6, 9, 14, 15, 20]. Recent works also consider visualization parameters to ensure perceptually similar visualizations and offer visualization-aware error guarantees [12]. Further, *progressive visualization* approaches perform computations in small, incremental steps, providing users with progressively improving approximate results [1, 16–19]. In contrast, we focus on implementing an adaptive indexing scheme for approximate query answering. *Adaptive indexing* techniques dynamically adjust indexes based on query workloads to optimize data access [2–5, 7, 11, 13, 21]. Compared to our work, these studies are based on exact query answering.

Recent techniques for AQP, which combine pre-computed aggregates and data sampling to facilitate interactive analysis, are similar to our work but in a different context [8, 15]. However, these approaches neither utilize adaptive indexing to support efficient query evaluation over large data files nor dynamically adjust the granularity of their pre-computed aggregates to accommodate the dynamic nature of data exploration scenarios. In contrast, our work leverages

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment. ISSN 2150-8097.

[1]The source code is available at *https://github.com/VisualFacts/RawVis*

an adaptive index and pre-computed aggregates to perform in-situ adaptive indexing with error-bound guarantees. Additionally, we allow for "partial" index adaptation to reduce associated costs while considering the user's accuracy constraints.

2 FRAMEWORK OVERVIEW

2.1 Exploration Model

In our scenario, we consider a user visually exploring data stored in a file (e.g., CSV file) using a 2D visualization technique (e.g., map, scatter plot), and analyzing it using visual tools (e.g., bar and line charts, heatmaps) as well as aggregate and statistical measures (e.g., mean, Pearson correlation) [10]. The data attributes may be numeric, spatiotemporal, categorical, or textual. At least two of these attributes must be numeric (e.g., longitude, latitude) and can be mapped to the X and Y axes of the 2D visualization (*axis attributes*).

Our framework is based on a formal *exploration model* that defines a set of exploratory and analytic operations to formulate user interactions [11]. It implements basic *exploration operations* over a 2D visualization plane, such as pan, zoom, filter, view object details, and selection of objects by defining 2D areas (i.e., range queries) over the visualized objects. Beyond exploratory operations, the model defines *analytic operations* that allow the user to visually analyze the objects by generating several types of visualizations that can aggregate, compare, or provide statistics on data properties.

2.2 Indexing Scheme

For the sake of simplicity, we describe the basic idea of our methods over the hierarchical tile-based VALINOR index (referred as *index*) [3], which is a simplified version of our VETI index [11].[2]

The index is stored in main memory and organizes the data objects into hierarchies of non-overlapping rectangle tiles. The index's tiles are defined over the domains of axis attributes. Each tile is associated with *metadata* (e.g., average, sum, count) that enables statistics computations. Particularly, metadata are numeric values calculated by algebraic aggregate functions over one or more attributes of the objects.

Consider the example where the 2D canvas is a map, the data points are hotels with ratings and their location on the map is given by the longitude and latitude values. Figure 1 (a) presents an example of the basic structure of an index, which divides the 2D space into 3×3 equally sized disjoint tiles ($t_1{\sim}t_4$), where the tile t_4 is further divided into 2×2 subtitles ($t_{4_a}{\sim}t_{4_d}$), forming a tile hierarchy[3]. Each tile has a range on the longitude and latitude attributes, contains the data points within that range and keeps an aggregate value (e.g., average rating) for the containing set.

Index Initialization. Initially, a "crude", lightweight initial version of the index is built, and progressively adjusts itself to the user interactions, by splitting the tiles visited into more fine-grained ones.

Index Adaptation. In what follows, we outline the index adaptation used for exact query answering. RawVis employs a progressive index adaptation technique that attempts to reduce the I/O operations and computations by adjusting the index based on the user interactions, e.g., exploration areas and required statistics. Adaptation is performed by modifying the structure of the index (e.g., tile size); and by enriching and updating its "information" (e.g., metadata). The adaptation method incrementally splits the tiles that overlap with a query into smaller subtiles. The splitting process considers factors related to I/O cost in order to decide whether to perform a split or not. Considering the locality-based characteristics of the exploration scenarios, tile splitting increases the likelihood that a future query will fully overlap a tile in the area, which the user exploration focuses on. The case of fully overlapped tiles allows the index to use the existing metadata, improving the query performance by reducing I/O operations on the file. In conjunction with the tile splitting, the index may be enriched by computing different metadata.

Figure 1 (b) shows the index adaptation when query Q is posed. The tiles t_1 and t_3 that overlap with Q are split to 2×2 subtiles. Note, for simplicity, we assume that there is no need to further split $t_{4_a}{\sim}t_{4_d}$ subtiles. So, in this case we have to reorganize the objects included in t_1 and t_3 tiles, and compute the metadata for each subtile.

[2]Compared to VALINOR, the VETI index combines tiles with tree structures enabling categorical-based aggregations, and supports resource-aware index management.

[3]For simplicity here we present equally sized tiles.

3 PARTIAL INDEX ADAPTATION FOR APPROXIMATE QUERY ANSWERING

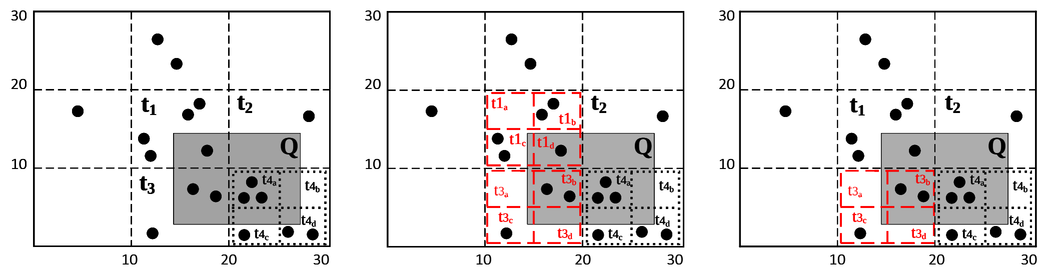
Our goal is to support approximate query evaluation over the index, reducing the cost of reading raw data from the file and the cost of adapting the index while ensuring that the results' accuracy meets a given constraint. While leveraging the index structure and the aggregate metadata can significantly reduce file accesses (i.e., execution time), there are cases such as the initialization phase and dense areas, where query execution exhibits lower performance.

The performance in the aforementioned cases become even more challenging when the user's exploratory queries involve attributes that are not directly stored in the index. Recall that, the index stores the axis attributes (e.g., longitude, latitude) used in 2D visual explorations. This enables efficiently determining which objects fall within the query window without direct access to the raw data file. However, queries involving analytic functions that utilize attributes not directly indexed (e.g., aggregating non-axis attributes) may necessitate file access to compute exact results.

The cases of exact and approximate query processing are illustrated in Figure 1. We assume that the index is already initialized. Figure 1(a) depicts the index before the query Q is posed, whereas Figure 1(b) and (c) depict the updated index after Q evaluation with 100% accuracy and the approximate case, respectively. Assume that the query Q requests an aggregate function (e.g., average rating of the hotels) to be computed over all objects within Q range.

Index Adaptation for Exact Query Answering Example. First, we identify the tiles that overlap with the query, i.e., t_1, t_2, t_3, and $t_{4_a} \sim t_{4_d}$. Next, we identify which of the overlapped tiles require file access. Tiles t_2 and $t_{4_b} \sim t_{4_d}$ are skipped, as they do not include any of the selected objects. Tile t_{4_a} contains objects, but it is fully contained in the query and, its indexed metadata can be used to compute the aggregates requested for Q, without additional data access.

However, tiles t_1 and t_3 are partially contained in the query, i.e., their metadata concerns the entire tile and not an aggregate value for the specific selected objects requested for Q. Thus, we need to read from the file the required attributes' values of the objects in t_1 and t_3 that are within the query. This results in reading three objects. Simultaneously, we split the tiles into four sub-tiles and compute and store metadata for the newly created sub-tiles t_{1_d} and t_{3_b}.



(a) Initial Index **(b) Adapted Index** **(c) Partially Adapted Index for Approximate Query Answering**

Figure 1: Index Adaptation Example (a) Initial index structure; (b) Exact query answering, splitting tiles t_1 and t_3; (c) Approximate query answering, splitting only t_3 and providing results within user accuracy constraints

3.1 Approach Overview

In this section we provide some basic concepts and the problem definition.

Query Confidence Interval. A key aspect is that for the partially contained tiles, we can compute (without accessing the file) the number of objects within the window query via the objects axis values stored in the index. Using the number of objects (*count*) along with the *sum*, *min*, and *max* aggregate metadata stored in each tile, we can deterministically bound the aggregate value of the objects in the query. This allows us to establish a *query confidence interval* and guarantee that the actual aggregate value falls within it. Based on that, we can approximate various aggregate functions, such as *sum*, *mean*, *min* and *max*.

For example, assume that we wish to compute the query confidence interval for the *sum* function over a non-axis attribute A for the objects within the window query Q. In this case, the *query confidence interval* for the *sum* function is calculated by using the precomputed metadata of each tile (i.e., objects count, minimum value, maximum value). Particularly, the *query confidence interval* is defined as:

$$\Bigg[\sum_{t\in\mathcal{T}_f} \mathrm{sum}_A(t) + \sum_{t\in\mathcal{T}_p} \mathrm{count}(t\cap Q)\cdot \mathrm{min}_A(t),$$
$$\sum_{t\in\mathcal{T}_f} \mathrm{sum}_A(t) + \sum_{t\in\mathcal{T}_p} \mathrm{count}(t\cap Q)\cdot \mathrm{max}_A(t)\Bigg],$$

where $\mathcal{T}_f$ denote the fully-contained tile set and $\mathcal{T}_p$ the partially-contained tiles set; $\mathrm{sum}_A(t)$, $\mathrm{min}_A(t)$ and $\mathrm{max}_A(t)$ the sum, the minimum and the maximum value of the attribute A in tile t, respectively; and $\mathrm{count}(t \cap Q)$ the number of objects inside the tile t that are selected by the query Q.

This computation can be generalized to the *mean*, *min* and *max*. For the *mean*, the confidence interval is determined by dividing the sum query confidence interval by the total count of objects included in the query. For the *min* and *max* aggregates, the query confidence interval is derived by considering the fully contained tiles' min or max values respectively and bounding the partially contained tiles' values within their min-max range.

Tile Confidence Interval. Following a similar approach as the query confidence interval, confidence intervals can be computed for partially-contained tiles. For example, for the *sum* aggregate function, given a query Q, a partially-contained tile t and a non-axis attribute A, the *tile confidence interval* for the *sum* over A is defined as: $[\mathrm{count}(t\cap Q)\cdot \mathrm{min}_A(t), \mathrm{count}(t\cap Q)\cdot \mathrm{max}_A(t)]$.

Approximate Value For an aggregate function included in the query, we compute an *approximate value* using: (1) the exact values from the query's fully contained tiles; and (2) approximate values from the query's partially contained tiles. For the partially contained tiles, the approximate values are derived using the tile's aggregate metadata. For example, the approximate value for the sum function is computed by multiplying each partially contained tile's mean value (derived from its min and max values) with the count of objects within the query range in that tile.

Upper Error Bound. By considering query confidence interval, we derive a relative *upper error bound*. The upper error bound is computed by normalizing the maximum difference between the approximate value computed and the query confidence interval bounds.

Problem Definition. Recall that, splitting is performed over the tiles that are partially contained within the query. When a split is performed we have to read from file the objects included in the partially contained tile and compute tile's metadata.

Let $\mathcal{T}$ be the *set of query's partially contained tiles*. Let $\mathsf{process}(t)$ be a function, which *processes a partially contained tile* $t \in \mathcal{T}$, i.e., splits the tile, reorganizes the objects in the subtiles, reads from the file the values of the objects included in t, and computes the metadata of each subtile. Let $\mathsf{process}(t).cost$ be the *time required to process the tile* t.

Finally, $\mathsf{error\_bound}(\mathcal{T},\mathcal{T}')$ denote *the upper error bound* between the query answers resulting from processing all the partially contained tiles $\mathcal{T}$ (exact answer) and a subset $\mathcal{T}' \subseteq \mathcal{T}$ (approximate answer).

Given the set of query's partially contained titles $\mathcal{T}$, our problem is to find a set $\mathcal{T}' \subseteq \mathcal{T}$ such that the cost of processing the tiles $\mathcal{T}'$ is minimized, and the answer error bound is lower or equal to a user-specified accuracy constraint ϕ. Formally:

$$\underset{\substack{\mathcal{T}'\\ \mathcal{T}'\subseteq\mathcal{T}}}{\arg\min} \sum_{\forall t\in\mathcal{T}'} \mathsf{process}(t).cost \ \text{s.t.}\ \mathsf{error\_bound}(\mathcal{T},\mathcal{T}') \le \phi$$

Processing Partially Contained Tiles. In order to find the subset of partially contained tiles which will process, we use the following simple approximation method.

For each partially contained tile $t \in \mathcal{T}$ we compute *a score* $s(t)$ that combines factors related to accuracy and processing cost. Particularly, for each t we consider: (1) *the width of the tile confidence interval* $w(t)$, that formulates the "degree of inaccuracy" of t. Note that, tiles with wider confidence intervals are considered more inaccurate; and (2) $\mathrm{count}(t \cap Q)$ that is *the number of objects inside t that are selected by the query* Q, that formulates the processing cost. Formally, the score $s(t)$ of a tile t is defined as: $s(t) = \alpha \cdot w(t) + (1-\alpha)/\mathrm{count}(t \cap Q)$, where $\alpha \in [0, 1]$ formulates the trade-off between the two metrics. Note that, $w(t)$ and $\mathrm{count}(t \cap Q)$ are normalized to $[0, 1]$.

We define a tiles selection policy that prioritizes the process of the tiles from $\mathcal{T}$ with the largest scores, progressively refining our results until they meet the user-specified accuracy constraint. So, the processed tiles correspond to tiles set $\mathcal{T}'$.

For example, in Figure 1(c) we depict a partially adapted index after evaluating the query Q. We have two partially contained tiles that include objects selected by the query, i.e., t_1 and t_3 Assume that the tile t_3 has larger score than t_1 So, we first process t_3, i.e., perform a splitting and we read from file the objects within tile. Then, if the estimated value for the query falls within the user-defined error bounds, there is no need to access the file for the objects in the tile t_1. So, the process of t_1 is avoided.

4 PRELIMINARY EVALUATION

In this section, we present a preliminary evaluation focusing on response time improvements achieved through partial index adaptation under different accuracy constraints. We used a sequence of queries and measured the evaluation time under 1% and 5% accuracy constraints compared to the exact query answering method. We used the synthetic dataset from [3, 11] with 10 numeric columns (11 GB). Each query, defined over two numeric attributes, specifies a window containing approximately 100K objects and is shifted 10~20% randomly to simulate a map-based exploration path. Finally, the score s for each tile used by the tile selection policy considers only the width of the tile confidence interval, i.e., $\alpha = 1$.

Figure 2 illustrates the evaluation times for a sequence of 50 *overlapping queries*. The black line represents the evaluation time for exact query answering, while the red and green lines represent the evaluation times for 1% and 5% max error bounds, respectively. The peaks in the evaluation times can be attributed to the exploration of previously unvisited areas, where the index is less refined.

The results indicate that our partial index adaptation approach can reduce query evaluation times, especially in the early stages of user interaction. The trade-off between accuracy and performance is evident, with higher accuracy constraints leading to slightly longer evaluation times but still outperforming the exact method.

The evaluation times closely follow the number of objects (i.e., CSV file rows) that need to be read from the raw data file. In the approximate cases, we can skip reading objects for some tiles partially overlapping the window query, while utilizing their aggregate metadata to bound their confidence interval. This is particularly effective for the first 20 queries where only a "crude" version of the index is available. As the index adapts for the exact evaluation scenario, the queries become comparable or slightly faster than the approximate ones because the index has been substantially adapted in the areas explored by the user, reducing the need to access the raw data file repeatedly. In contrast, the approximate cases may result in less efficient query evaluation as the index is not as thoroughly adapted.

Overall, the results seem promising, with most queries being evaluated faster for the approximate cases, especially in the first queries of the exploration scenario. For example, at query 20, the 5% accuracy method performs considerable well, being 4× faster than the exact method, while the 1% accuracy method is about twice as fast. However, in some queries, the approximate methods can result in higher evaluation times because the exact method has progressively refined the index more, allowing for quicker evaluations. This uncommon behavior highlights the trade-off between partial and full index adaptation, which we aim to improve by developing advanced tile selection policies and enabling more index adaptation even if the accuracy constraints have been satisfied. Overall, considering the whole exploration scenario, the 5% and 1% methods are about 40% and 30% faster, respectively.

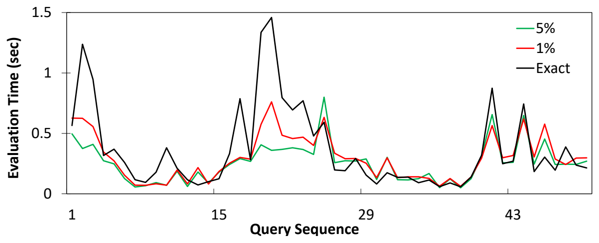


Figure 2: Evaluation Time for Different Error Bounds

ACKNOWLEDGMENTS. This work was supported by the ExtremeXP project (EU Horizon program, GA 101093164).

REFERENCES

[1] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. Blinkdb: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In *EuroSys*.
[2] Ioannis Alagiannis, Renata Borovica, Miguel Branco, Stratos Idreos, and Anastasia Ailamaki. 2012. Nodb: Efficient Query Execution on Raw Data Files. In *SIGMOD*.
[3] Nikos Bikakis, Stavros Maroulis, George Papastefanatos, and Panos Vassiliadis. 2021. In-situ Visual Exploration over Big Raw Data. *Information Systems* (2021).
[4] Saheli Ghosh, Ahmed Eldawy, and Shipra Jais. 2019. AID: An Adaptive Image Data Index for Interactive Multilevel Visualization. In *ICDE*.
[5] Pedro Holanda and Stefan Manegold. 2021. Progressive Mergesort: Merging Batches of Appends into Progressive Indexes. In *EDBT*.
[6] Albert Kim, Eric Blais, Aditya G. Parameswaran, Piotr Indyk, Samuel Madden, and Ronitt Rubinfeld. 2015. Rapid Sampling for Visualizations with Ordering Guarantees. *PVLDB* (2015).
[7] Konstantinos Lampropoulos, Fatemeh Zardbani, Nikos Mamoulis, and Panagiotis Karras. 2023. Adaptive Indexing in High-Dimensional Metric Spaces. *PVLDB* (2023).
[8] Xi Liang, Stavros Sintos, Zechao Shang, and Sanjay Krishnan. 2021. Combining Aggregation and Sampling (Nearly) Optimally for Approximate Query Processing. In *SIGMOD*.
[9] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. 2013. imMens: Real-time visual querying of big data. In *Computer graphics forum*.
[10] Stavros Maroulis, Nikos Bikakis, George Papastefanatos, Panos Vassiliadis, and Yannis Vassiliou. 2021. RawVis: A System for Efficient In-situ Visual Analytics. In *SIGMOD*.
[11] Stavros Maroulis, Nikos Bikakis, George Papastefanatos, Panos Vassiliadis, and Yannis Vassiliou. 2022. Resource-Aware Adaptive Indexing for In-situ Visual Exploration and Analytics. *VLDBJ* (2022).
[12] Stavros Maroulis, Vassilis Stamatopoulos, George Papastefanatos, and Manolis Terrovitis. 2024. Visualization-Aware Time Series Min-Max Caching with Error Bound Guarantees. *PVLDB* (2024).
[13] Matthaios Olma, Manos Karpathiotakis, Ioannis Alagiannis, Manos Athanassoulis, and Anastasia Ailamaki. 2020. Adaptive partitioning and indexing for in situ query processing. *The VLDB Journal* 29 (01 2020).
[14] Yongjoo Park, Michael J. Cafarella, and Barzan Mozafari. 2016. Visualization-aware sampling for very large databases. In *ICDE*.
[15] Jinglin Peng, Dongxiang Zhang, Jiannan Wang, and Jian Pei. 2018. AQP++: Connecting Approximate Query Processing With Aggregate Precomputation for Interactive Analytics. In *SIGMOD*.
[16] Sajjadur Rahman, Maryam Aliakbarpour, Hidy Kong, Eric Blais, Karrie Karahalios, Aditya G. Parameswaran, and Ronitt Rubinfeld. 2017. I've Seen "Enough": Incrementally Improving Visualizations to Support Rapid Decision Making. *PVLDB* (2017).
[17] Zeyuan Shang, Emanuel Zgraggen, Benedetto Buratti, Philipp Eichmann, Navid Karimeddiny, Charlie Meyer, Wesley Runnels, and Tim Kraska. 2021. Davos: a system for interactive data-driven decision making. *PVLDB* (2021).
[18] Alex Ulmer, Marco Angelini, Jean-Daniel Fekete, Jörn Kohlhammer, and Thorsten May. 2024. A Survey on Progressive Visualization. *IEEE TVCG* (2024).
[19] Yunhai Wang, Yuchun Wang, Xin Chen, Yue Zhao, Fan Zhang, Eugene Wu, Chi-Wing Fu, and Xiaohui Yu. 2023. OM3: An Ordered Multi-level Min-Max Representation for Interactive Progressive Visualization of Time Series. *SIGMOD* (2023).
[20] Jia Yu and Mohamed Sarwat. 2020. Turbocharging Geospatial Visualization Dashboards via a Materialized Sampling Cube Approach. In *ICDE*.
[21] Fatemeh Zardbani, Nikos Mamoulis, Stratos Idreos, and Panagiotis Karras. 2023. Adaptive Indexing of Objects with Spatial Extent. *PVLDB* (2023).