Time-Series Data Management and Analytics: A Survey

www.surveyx.cn

Abstract

Time-series data, characterized by sequences of data points indexed in time order, is pivotal in domains such as finance, healthcare, and IoT, necessitating specialized databases like InfluxDB and Prometheus for efficient management. These databases address the unique demands of time-series data through techniques such as downsampling, time windowing, and high-cardinality indexing, facilitating high-performance querying and storage. This survey provides a comprehensive overview of time-series data management, emphasizing the role of Time-Series Databases (TSDBs) in supporting real-time analytics and stream processing. Key techniques such as data aggregation, anomaly detection, and privacy preservation are explored, highlighting their significance in optimizing storage and query performance. The paper also delves into the scalability challenges faced by TSDBs, proposing architectural strategies and advanced techniques to enhance performance in handling growing data volumes. The survey concludes by summarizing advancements in the field and identifying future research opportunities, including the integration of machine learning models and the development of hybrid approaches for improved trend prediction. These insights underscore the critical role of TSDBs in modern analytics, enabling timely and accurate data-driven decision-making across various applications.

1 Introduction

1.1 Significance of Time-Series Data

Time-series data has become essential in modern analytics due to the exponential increase in data generated across diverse fields such as finance, healthcare, technology, and behavioral analytics. In finance, it underpins forecasting and decision-making, enabling precise predictions and strategic planning. Similarly, in healthcare, time-series data is vital for analyzing multivariate clinical data, necessitating advanced methods to manage its irregularities [1].

The rise of the Internet of Things (IoT) has amplified the need for effective time-series data processing, as IoT devices generate vast data streams continuously [2]. In cyber-physical systems (CPS), analyzing time-series data is crucial for addressing data deluge challenges, highlighting its growing significance in contemporary analytics [3].

Effective visualization of time-series data is critical for trend identification and insight extraction; however, the increasing dataset sizes pose significant challenges [4]. The demand for enhanced interpretability of deep learning models in time-series analysis further emphasizes the expanding role of this data type across various sectors [1].

In multimedia contexts, the efficient archiving and analysis of large-scale social media content, particularly regarding temporal and spatial dimensions, illustrate the contemporary relevance of time-series data [5]. Moreover, the influence of distinct visualization techniques on interpreting and predicting public health data, such as COVID-19 case counts, underscores the critical role of

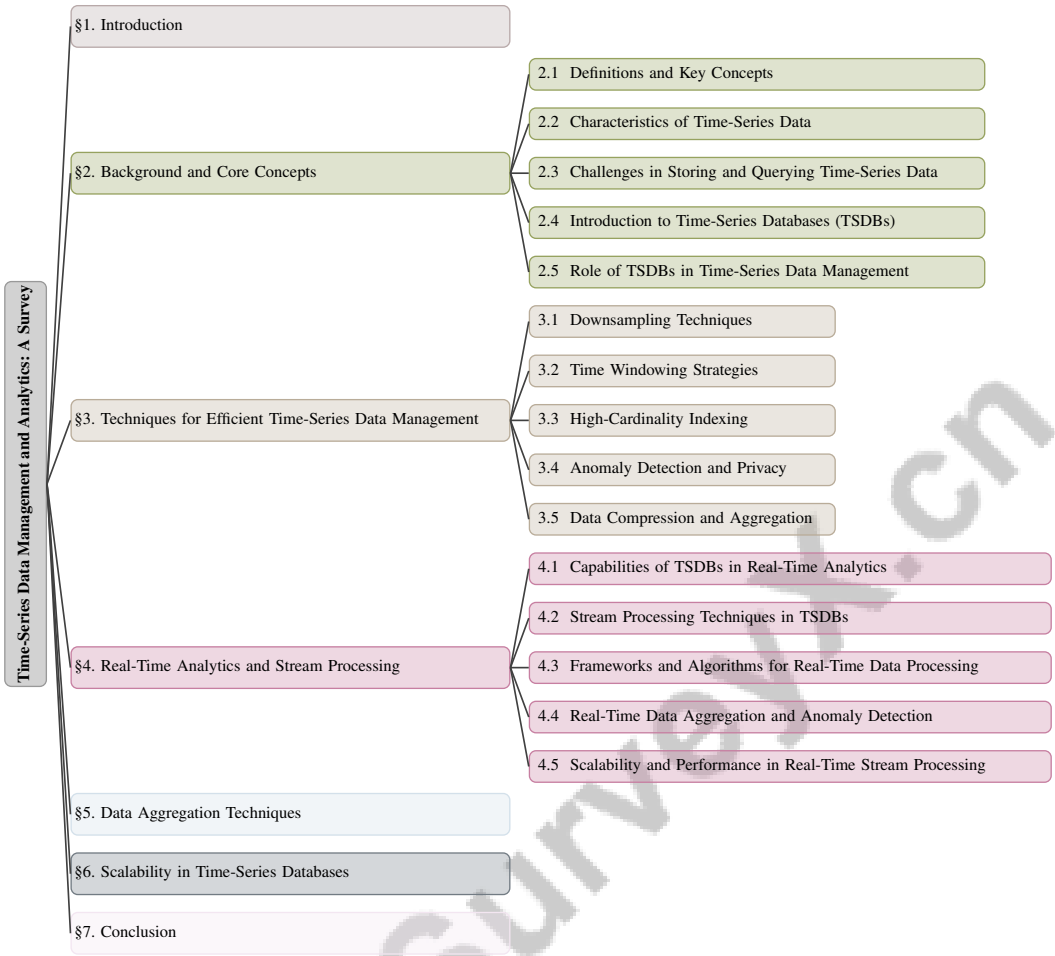

Figure 1: chapter structure

time-series data in public health analytics [6]. As businesses increasingly depend on real-time data analysis and management, the importance of time-series data continues to grow, solidifying its foundational role in modern analytics [7].

1.2 Specialized Databases: InfluxDB and Prometheus

InfluxDB and Prometheus exemplify specialized databases tailored to meet the unique demands of time-series data management. These databases are optimized to efficiently handle the high frequency and volume of updates typical of time-series data, which conventional relational databases often struggle to manage effectively [8]. InfluxDB, developed by InfluxData, is recognized for its high-performance capabilities in storing and querying time-series data, featuring time-based data retention policies and continuous queries that facilitate real-time analytics. Conversely, Prometheus, an open-source monitoring solution by SoundCloud, emphasizes a powerful query language and a multi-dimensional data model that supports high-cardinality indexing. Both databases utilize techniques like downsampling and time windowing to enhance storage and query performance, ensuring scalability and efficiency in managing large-scale time-series datasets. Their development and benchmarking highlight their critical role in modern analytics, particularly in environments where timely and accurate data insights are essential [9].

1.3 Structure of the Survey

This survey is structured to provide a comprehensive overview of time-series data management and analytics, focusing on specialized databases such as InfluxDB and Prometheus. The introduction establishes the significance of time-series data across various domains and highlights the critical

role of databases designed for such data. It delineates essential techniques, including downsampling, time windowing, and high-cardinality indexing, which are crucial for efficient time-series data management. Specifically, downsampling facilitates data point aggregation or selection to enhance visualization and analysis, while time windowing enables focused analysis over specific intervals, and high-cardinality indexing supports quick access to diverse data attributes, optimizing performance in handling large-scale datasets [10, 11, 12, 13, 4].

The second section delves into the background and core concepts, providing definitions and explanations of essential terms related to time-series data. It discusses the unique characteristics of time-series data, the challenges in storing and querying it, and introduces time-series databases (TSDBs) as effective solutions to these challenges.

The third section explores various techniques employed by TSDBs for efficient data management, including downsampling, time windowing strategies, and high-cardinality indexing, as well as methods for anomaly detection, privacy, data compression, and aggregation techniques.

Section four focuses on TSDB capabilities in supporting real-time analytics and stream processing, discussing how these databases manage continuous data flows to enable immediate insights and exploring integrated stream processing techniques. It also covers frameworks and algorithms for real-time data processing, aggregation, anomaly detection, and the scalability and performance challenges in real-time stream processing.

The fifth section is dedicated to data aggregation techniques used in TSDBs, providing an overview of common methods and specific aggregation techniques such as delta summation and CRTS. It discusses symbolic and compression techniques, clustering, and spectrogram-based aggregation, alongside advanced techniques that enhance scalability and performance.

The sixth section analyzes scalability challenges faced by TSDBs as data volumes grow, exploring architectural strategies and specific techniques to improve scalability.

The survey concludes by summarizing key findings and advancements in time-series data management and analytics, discussing future directions and potential research opportunities, including the evaluation of deep neural network models for trend prediction in time-series data and comparing hybrid approaches like TreNet against traditional machine learning algorithms [14].The following sections are organized as shown in Figure 1.

2 Background and Core Concepts

2.1 Definitions and Key Concepts

Time-series data management encompasses a variety of concepts and methodologies essential for effective analysis across multiple domains. Time Series Classification (TSC) is pivotal for pattern detection in finance, healthcare, and environmental monitoring [15]. As time-series databases grow, effective representation methods become increasingly vital, necessitating the capture of intrinsic characteristics for diverse analytical tasks [16]. Symbolic Aggregate approXimation (SAX) and Piecewise Linear Approximation (PLA) are significant for enhancing data representation accuracy and compressing large datasets, respectively [17]. State Space Models (SSMs) offer robust frameworks for forecasting continuous systems by predicting future states from observed data [18]. Change-point analysis is crucial for timely detection of significant shifts in data patterns, such as changes in mean or variance [19]. Platforms designed for effective visualization and interpretation capture changes in frequency, sentiment, and topic distribution over time [6]. Generating natural language summaries that mimic human summarization processes enhances accessibility for non-specialists [1].

Large language models (LLMs) have been re-evaluated against simpler models for forecasting, classification, and anomaly detection, showing the potential of advanced machine learning techniques to improve time-series analysis [14]. Tokenized time-series models are emerging to address inefficiencies, promoting models that generalize across tasks and domains [20]. Challenges persist in learning effective representations when expert features are available but conventional transformations are inapplicable [16]. Predicting price movements in low-frequency financial data is complicated by noise and non-linear market interactions [18]. Learning distributions from irregularly-sampled data and detecting periodic patterns without labeled samples remain complex issues. Traditional bootstrapping methods often fail to respect temporal correlations inherent in time-series data [16]. Discovering

significant subsequences, such as in electrocardiogram data, involves identifying minimum-length contiguous subsequences whose removal alters classification outcomes, highlighting the complexities of time-series analysis. Outlier detection techniques are crucial for unsupervised anomaly identification [19].

Explainability in time-series classification models must overcome the limitations of existing methods, which often focus on time-domain features. Analyzing multivariate time series data, characterized by serial correlation and cross-sectional dependencies, requires refined mapping methods [15]. Understanding logical structures within extensive time-series data generated by Cyber-Physical Systems (CPS) is essential for enhancing system performance [3]. Effective reasoning mechanisms within stream processing systems are necessary for handling temporal data and deriving insights from past and future events [21]. Regression analysis in time series, particularly predicting water levels influenced by rainfall and climate variables, remains a significant research area [22]. Foundational definitions and concepts in time-series data management are critical for developing advanced analytical techniques, such as persistent homology and matrix profile algorithms, which effectively tackle the complexities of time-series data, including high-velocity data streams from IoT devices and anomaly detection in diverse applications ranging from healthcare to finance [23, 24, 25, 17, 26].

2.2 Characteristics of Time-Series Data

Time-series data is inherently sequential, with each data point indexed in time, facilitating the analysis of temporal patterns and dependencies absent in other data types. This temporal structure is vital for understanding dynamic processes across finance, healthcare, and environmental science, enabling predictive modeling and informed decision-making [27]. The multidimensional and complex nature of time-series data often necessitates sophisticated analytical methods to extract meaningful insights. Traditional clustering techniques may falter when addressing the intricacies of time-series and curves, which often involve multiple dimensions and complex interrelationships [28]. The challenge of effectively clustering such data highlights the need for advanced methodologies to accommodate inherent complexities.

Time-series data typically consists of multiple segments, each exhibiting distinct patterns that significantly impact classification accuracy. Retaining segment-level information is essential for enhancing classification precision, as it fosters a nuanced understanding of the underlying data structure [29]. Additionally, the presence of temporal dependencies means that query responses may rely on data not yet received or on data from the distant past, necessitating systems to maintain extensive input histories [21]. Frameworks like EA-ConvNets, designed for early classification of time series, underscore the importance of learning discriminative shapelets that capture essential characteristics at multiple scales [27]. These frameworks illustrate the need for specialized analytical techniques to effectively manage the unique challenges posed by time-series data, including temporal dependencies and multidimensionality.

2.3 Challenges in Storing and Querying Time-Series Data

Efficient management of time-series data presents significant challenges, particularly regarding storage and querying, due to its high dimensionality, variability, and real-time processing requirements. A primary challenge is the inadequacy of existing methods to effectively track and analyze data change patterns, often resulting in plausible yet incorrect responses [30]. This issue is compounded by the inefficiency of current methods for performing similarity queries, especially when approximate matching is required [13]. Scalability and reliability are critical concerns, as traditional aggregation methods struggle with the large volumes of sensor data generated in real-time scenarios [31]. Extracting hierarchical features from non-stationary time series, which exhibit multiple quasi-stationary characteristics, adds complexity, particularly in the absence of prior knowledge about unique behaviors [32].

Reliance on synthetic data in benchmarks limits their applicability to real-world scenarios, failing to capture practical complexities [2]. The lack of interpretability in neural representations complicates their use in critical decision-making tasks, as these models often do not yield meaningful insights into the underlying data [33]. Clustering time-series data is particularly challenging due to high dimensionality, noise, and the absence of prior knowledge about classes, complicating the clustering process [28]. Existing methods often require known periodicity and modes, which are typically

unknown in real-world scenarios, hindering accurate behavior pattern identification [34]. Furthermore, benchmarks focusing on batch processing do not meet the needs for real-time analytics, leading to delays and inefficiencies in data handling [35]. The limitations of existing clustering algorithms, such as dynamic time warping (DTW), in learning logical structures within time-series data result in suboptimal groupings [3]. Pooling methods like global average pooling (GAP) and global max pooling (GMP) discard the temporal position of features, leading to position-invariant representations that degrade classification performance [29]. Capturing complex relationships between predictor variables and outcomes, particularly non-linearities and long-term dependencies, remains a formidable challenge in time-series data analysis [22]. These challenges underscore the necessity for innovative approaches and advanced technologies to enhance the storage, querying, and overall management of time-series data, ensuring adaptability to the evolving demands of modern data environments.

2.4 Introduction to Time-Series Databases (TSDBs)

Time-series databases (TSDBs) are specialized systems designed to meet the unique requirements of time-series data, characterized by sequential data points indexed in time. TSDBs play a vital role in sectors such as finance, healthcare, and Internet of Things (IoT) applications, where efficient storage, querying, and analysis of large volumes of time-series data are essential for monitoring and decision-making. The surge in IoT devices generating massive time-series data has increased the demand for robust TSDBs. However, selecting the most suitable TSDB can be challenging due to varying performance benchmarks often reliant on synthetic data that may not accurately reflect real-world scenarios. The development of semantic models for monitoring data streams enhances TSDB capabilities, enabling effective data management and retrieval across complex IoT-Edge-Cloud infrastructures, facilitating timely insights and informed decision-making in critical applications [36, 2]. TSDBs are adept at managing the dynamic and real-time processing demands of time-series data, ensuring high-performance analytics and continuous data flow management.

A fundamental feature of TSDBs is their ability to handle high-frequency data, essential for applications requiring real-time insights and decision-making, such as monitoring and forecasting in financial markets. Advanced data mining tools like the matrix profile significantly enhance time-series analysis by enabling efficient identification of patterns and anomalies. The matrix profile serves as a versatile, parameter-free data structure facilitating intra- and inter-time series similarity joins, allowing for the discovery of conserved structures and anomaly detection across diverse applications, including motif discovery and semantic segmentation. Its capability to process both single and multidimensional time series makes it a powerful tool across fields, from seismology to medical diagnostics, ultimately improving interpretability and accuracy without extensive training datasets [24, 37, 25, 26]. Additionally, TSDBs employ sophisticated compression techniques to optimize storage and enhance query performance, such as erasing the least significant bits of floating-point values for efficient data retrieval.

Moreover, TSDBs are pivotal in anomaly detection, a critical aspect of time-series data management. Techniques like the WATCH method, which utilizes the Wasserstein distance for change point detection, and permutation entropy-based methods for local mixing detection, are integral to maintaining data integrity and enhancing analytical outcomes. Frameworks such as HSDF, which models non-stationary time series data using Probabilistic Finite State Automata, further exemplify TSDBs' advanced capabilities in managing complex data behaviors [32].

Scalability is a critical concern for TSDBs, which must handle increasing data volumes while maintaining performance. Benchmarking suites that comprehensively evaluate TSDB performance using real datasets from various domains highlight the importance of scalability in time-series data management [2]. Additionally, extending standard SQL to support streaming queries facilitates real-time analytics, allowing developers to implement these capabilities without needing to learn new complex languages [38].

2.5 Role of TSDBs in Time-Series Data Management

Time-series databases (TSDBs) are essential in addressing the complex challenges of time-series data management, particularly in environments characterized by high data velocity and volume. They are specifically engineered to efficiently manage the sequential and often high-frequency nature of time-series data, critical in finance, healthcare, and IoT applications. By leveraging specialized

data structures and algorithms, TSDBs provide robust solutions for storing, querying, and analyzing time-series data, ensuring real-time insights can be derived [39].

One primary challenge in time-series data management is the need for scalable and accurate data classification and analysis. TS-CHIEF, a scalable ensemble algorithm, exemplifies how TSDBs can integrate advanced machine learning techniques to enhance time-series data classification. Utilizing tree classifiers and various embedding techniques, TS-CHIEF achieves high accuracy, demonstrating TSDBs' potential to support complex analytical tasks with scalable performance [40]. TSDBs also address benchmarking and performance evaluation challenges, crucial for optimizing database operations under diverse workloads. The IoTDB-Benchmark serves as a standard for evaluating TSDB systems, providing a framework for effective comparison across databases. This benchmarking process is vital for understanding TSDB performance under various conditions, facilitating improvements in design and implementation [39].

In addition to scalability and benchmarking, TSDBs incorporate advanced compression techniques to optimize storage and query performance. These techniques are essential for managing substantial volumes of data generated by real-time applications, enhancing data accessibility and usability. Frameworks like the Tempo-Spatial Content Delivery Network (TS-CDN) optimize data storage and retrieval through hash functions, significantly reducing redundancy and ensuring unique data archives. Platforms such as AlertMix cater to the growing demand for immediate processing of streaming data from diverse sources, addressing critical business needs in areas like trading and fraud detection. As traditional batch processing methods, exemplified by Hadoop, fall short in handling real-time and interactive queries, developing specialized solutions for non-batch workloads becomes increasingly vital for maintaining actionable insights from big data streams [41, 42, 43]. Furthermore, TSDBs enhance anomaly detection capabilities, employing sophisticated methods such as change point detection and permutation entropy-based techniques to maintain data integrity and improve analytical outcomes.

Time Series Databases (TSDBs) are essential for effective time-series data management, specifically designed to handle the unique challenges posed by the exponential growth of time-series data generated by applications such as the Internet of Things (IoT) and large-scale Cyber-Physical Systems (CPSs). TSDBs offer tailored solutions that enhance the storage, querying, and analysis of vast volumes of time-series data, leveraging specialized architectures and functionalities that outperform traditional Database Management Systems (DBMSs). Their capabilities include efficient data injection and query execution, as well as support for advanced features like Stream Processing and Approximate Query Processing (AQP), making them indispensable for both academic and industrial applications [23, 44, 2]. Their continuous evolution and integration of cutting-edge technologies underscore their critical contribution to modern data analytics, enabling timely and accurate decision-making across various domains.

3 Techniques for Efficient Time-Series Data Management

Category	Feature	Method
Downsampling Techniques	Frequency-Based Transformation	SQTSD[13]
Time Windowing Strategies	Multi-Scale Analysis Representation Transformation	MSCNN[45] LDVR[46]
High-Cardinality Indexing	Preselection Techniques Time-Frequency Analysis Representative Clustering	MMLTTB[47] ViT-num-spec[48] MTp[34]
Anomaly Detection and Privacy	High-Dimensional and Temporal Detection Influence and Adjustment Techniques	WATCH[49], STDN[50] MQM[51], FTSA[17]
Data Compression and Aggregation	Time-Series Analysis Optimized Clustering Techniques Symbolic and Semi-Supervised Methods	SpectralX[52], MHVG[53], PE[54] MLSDA[55], DTW-C++[56] SuSL4TS[57], ISNR[33]

Table 1: The table summarizes various methods employed in efficient time-series data management, categorized into downsampling techniques, time windowing strategies, high-cardinality indexing, anomaly detection and privacy, and data compression and aggregation. Each category highlights specific features and methods, referencing key studies that contribute to advancements in the field. This comprehensive overview aids in understanding the diverse approaches used to address the challenges of managing large-scale time-series data.

Time-series data management is increasingly challenged by the growing volume and velocity of data. Downsampling emerges as a fundamental technique to address these challenges by reducing data resolution while preserving essential information. This is crucial in environments with continuous data generation and high-frequency updates, optimizing storage and processing capabilities. Table 3 presents a detailed classification of methods utilized in time-series data management, showcasing the diversity and effectiveness of techniques across different domains. The following subsection delves into various downsampling methodologies and their applications in contemporary data environments.

3.1 Downsampling Techniques

Downsampling is critical for conserving storage and maintaining crucial information in time-series data management. Techniques like the DStream framework maintain a fixed-capacity subsample, balancing coverage across temporal criteria, while libraries such as tsdownsample utilize CPU optimizations for rapid processing, supporting scalable visualizations [58, 4, 10, 47]. These methods address large dataset challenges, enhancing pattern recognition and insight extraction across domains. Transforming time-series data into the frequency domain and applying linear transformations have improved querying efficiency of downsampled data [13]. Techniques like permutation entropy quantify predictability, underscoring downsampling's role in real-time data processing.

As illustrated in Figure 2, key downsampling techniques in time-series data management are categorized into frameworks and libraries, advanced techniques, and real-time processing methods, highlighting their applications and innovations. Efficient downsampling techniques are indispensable for managing extensive datasets, enabling responsive visualizations. Advanced stream processing techniques enhance system responsiveness, supporting real-time analysis and insight extraction from continuous data flows [45, 59, 60, 61, 43]. Downsampling techniques, such as the MinMaxLTTB algorithm, position themselves at the forefront of data management strategies, enabling effective handling of large time-series volumes through fast, memory-efficient processing [58, 10, 47, 23].

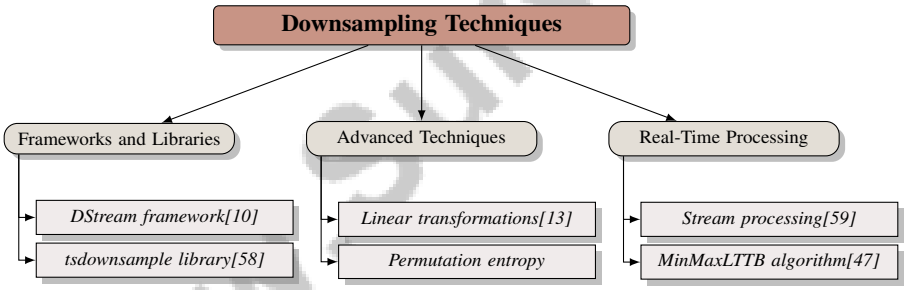

Figure 2: This figure illustrates key downsampling techniques in time-series data management, categorizing them into frameworks and libraries, advanced techniques, and real-time processing methods, highlighting their applications and innovations.

3.2 Time Windowing Strategies

Time windowing segments continuous data streams into discrete intervals, facilitating efficient processing, analysis, and storage. This strategy is vital in high-volume environments where rapid data arrival necessitates structured data for analytical operations like aggregation and anomaly detection. Techniques such as Progressive Window Widening adaptively adjust time intervals to capture varying pattern durations, enhancing streaming data queries [62, 63, 64].

Managing window overlaps ensures accurate processing, while Progressive Window Widening uses sliding windows to promptly recognize patterns [64, 63]. Multi-scale feature leverage, demonstrated by MSCNN, improves classification by providing a comprehensive understanding of input data [45]. Visualizing data within specific windows enhances trend interpretation, as seen in COVID-19 data analysis [65]. Transforming 1-D time-series data into 2-D images enables the use of pre-trained 2-D CNN models, improving representation learning [46].

Time windowing strategies are essential for segmenting time-series data, enhancing pattern and trend detection while ensuring data integrity by minimizing noise. Techniques like Progressive Window Widening and Zoom-SVD optimize analysis by capturing key patterns and smoothing noise,

thus enhancing decision-making in applications like computer security and user behavior tracking [66, 67, 68, 11, 63].

3.3 High-Cardinality Indexing

Method Name	Indexing Techniques	Scalability Solutions	Application Domains
MMLTTB[47]	Minmax Algorithm	Minmax Preselection Method	Time Series Visualization
ViT-num-spec[48]	-	Vision Transformers	Financial Time Series
MTp[34]	Segment Tree	Exemplar-based Clustering	Time-series Data

Table 2: Overview of advanced methods for high-cardinality indexing in time-series databases, detailing their indexing techniques, scalability solutions, and application domains. The table highlights the MinMaxLTTB algorithm, vision transformers for time-frequency spectrograms, and exemplar-based clustering mechanisms, demonstrating their effectiveness in various time-series data applications.

High-cardinality indexing is crucial in time-series databases (TSDBs) for efficient querying of numerous unique data points, especially in high-velocity environments where rapid data access is vital. Advanced techniques like probabilistic models and sampling methods capture data uncertainty, enhancing processing efficiency [69, 45, 38, 42, 61]. High-cardinality indexing structures facilitate rapid data retrieval and analysis. Table 2 provides a comprehensive summary of innovative methods employed for high-cardinality indexing in time-series databases, emphasizing their indexing strategies, scalability approaches, and relevant application domains.

Implementations in TSDBs like InfluxDB and TimescaleDB showcase their ability to process large-scale data efficiently in high-cardinality scenarios [39]. Innovative methods, such as the MinMaxLTTB algorithm, enhance scalability by reducing time complexity [47]. Time-frequency spectrograms with vision transformers offer novel perspectives, capturing both time and frequency information [48]. Exemplar-based clustering mechanisms address temporal variability and uncertainty, minimizing the number of clusters [34].

High-cardinality indexing is vital for efficient querying and analysis of large numbers of unique time-series data points. Employing sophisticated indexing techniques ensures TSDBs effectively manage high-cardinality environments, supporting scalable, high-performance solutions for diverse applications, including IoT and high-frequency financial data analysis [39, 8, 70, 2, 44].

3.4 Anomaly Detection and Privacy

Anomaly detection in time-series data identifies deviations from expected patterns, signaling events such as fraud or system failures. Advanced methodologies enhance outlier detection across various fields. FTSA uses influence vectors to adjust persistent homology calculations, improving robustness [17]. The WATCH method models high-dimensional data distribution for change detection, suitable for real-time applications [49].

Integrating time-frequency domain analysis, as in SpectralX, enhances classifier interpretability through time-frequency representations [52]. Statistical frameworks detect changes in dynamics, capturing temporal relationships indicative of nonstationarity [50].

Privacy preservation in time-series data management requires robust mechanisms to secure sensitive information. The Markov Quilt Mechanism ensures Pufferfish privacy by adding noise based on correlation among nearby nodes, akin to differential privacy [51]. This approach balances privacy with data utility, essential for applications managing sensitive data.

Fault tolerance in stream processing systems enhances reliability and efficiency, ensuring anomaly detection systems remain operational amid faults [5]. Future research focuses on real-time data cleaning tools and adaptive algorithms, supporting secure and reliable time-series analysis [35].

Effective time-series data management integrates advanced anomaly detection techniques with robust privacy-preserving mechanisms. These include differential and inferential privacy, crucial for safeguarding sensitive information in IoT and cloud services. Systems like TimeCrypt facilitate secure analytics over encrypted data, enhancing utility while maintaining confidentiality. A comprehensive approach combining detection techniques with privacy protections is essential for optimizing integrity and security in time-series management [23, 51, 71, 19, 26].

3.5 Data Compression and Aggregation

Data compression and aggregation are crucial for efficient time-series data management, optimizing storage and query performance while maintaining integrity. Techniques like Piecewise Linear Approximation (PLA) balance compression, latency, and error metrics in streaming environments [72]. Advanced methods, such as Short-Time Fourier Transform (STFT) with perturbation techniques, enhance feature identification within time-series data [52]. The MHVG method preserves dependencies between time series components through a multilayer structure [53].

Efficient dynamic time warping methods reduce memory usage and computational time, crucial for large datasets [56]. However, traditional techniques like classic DTW may provide superior accuracy despite innovative methods' computational advantages [68].

Discovering important subsequences in time-series data, such as electrocardiogram readings, involves optimized algorithms for efficient detection and relevance quantification [55]. Calculating permutation entropy at varying resolutions detects local mixing effects, offering nuanced insights [54].

The ISNR model employs an encoder-decoder structure with vector quantization for symbolic representations, enhancing interpretability [33]. Integrating semi-supervised and unsupervised learning, as demonstrated by SuSL4TS, allows effective classification without extensive feature extraction [57].

Data compression and aggregation techniques are essential for managing time-series data, enhancing storage efficiency and retrieval processes while maintaining integrity and utility. Advanced methods like adaptive Brownian bridge-based aggregation (fABBA) and modified SAX optimize dimensionality reduction and improve accuracy in data representation [62, 73, 74]. Lightweight compression methods leveraging statistical approaches offer superior compression ratios, suitable for resource-constrained environments like IoT applications. These advancements support effective handling of large-scale time-series datasets, providing timely and accurate insights across diverse applications.

Feature	Downsampling Techniques	Time Windowing Strategies	High-Cardinality Indexing
Optimization Focus	Data Resolution	Interval Segmentation	Data Retrieval
Processing Efficiency	Rapid Processing	Efficient Analysis	Rapid Querying
Application Domain	Pattern Recognition	Trend Detection	Tsdb Management

Table 3: This table provides a comparative analysis of three key methodologies employed in time-series data management: downsampling techniques, time windowing strategies, and high-cardinality indexing. Each method is evaluated based on its optimization focus, processing efficiency, and application domain, highlighting their distinct roles and contributions to efficient data handling.

4 Real-Time Analytics and Stream Processing

4.1 Capabilities of TSDBs in Real-Time Analytics

Time-series databases (TSDBs) play a crucial role in real-time analytics by efficiently managing continuous data streams from sources like IoT and CPSs. These databases address the limitations of traditional DBMSs, offering optimized data ingestion and query execution processes essential for rapid decision-making in dynamic environments [23, 39, 2]. TSDBs such as TimescaleDB, MonetDB, ExtremeDB, and Kairos-H2 leverage specialized data structures and algorithms to enhance storage, querying, and processing efficiency, thereby improving analytical performance across diverse applications.

A notable feature of TSDBs is their capability to model temporal dependencies, with techniques like EA-ConvNets significantly enhancing early time-series classification accuracy, crucial for prompt predictions [27]. The HSDF algorithm further aids in feature extraction from non-stationary time series, offering low computational complexity suitable for real-time applications [32].

As illustrated in Figure 3, the capabilities of TSDBs in real-time analytics can be categorized into three primary areas: optimized data management, advanced processing frameworks, and innovative analytical techniques. Each of these categories is supported by specific methodologies and technologies that enhance the performance and efficiency of TSDBs in handling real-time data streams. The

integration of advanced processing frameworks, particularly SQL-based streaming query models, empowers developers to build robust real-time data processing applications, improving turnaround times and accessibility for non-technical users [35]. Systems like Spark Streaming exemplify the ability to handle large datasets with reduced latency compared to traditional batch processing, underscoring the importance of real-time stream processing in modern data environments [35].

Innovative methodologies, such as MTpattern, capture human behavior patterns from uncertain temporal data, providing immediate insights [34]. Real-time vehicle data processing benchmarks ensure timely notifications for drivers based on live data, highlighting TSDBs' practical applications [75].

TSDBs are indispensable for real-time analytics, offering tools and methodologies to manage continuous data flows and deliver immediate insights. Their integration of advanced processing frameworks, efficient data management solutions, and innovative analytical techniques positions them as leaders in real-time data management, essential for applications in trading, fraud detection, system monitoring, and social media analytics. A unified streaming architecture combining push-based and pull-based mechanisms enhances processing efficiency, reduces latency, and increases throughput, facilitating seamless handling of multi-source streaming data in real-world scenarios [61, 43].

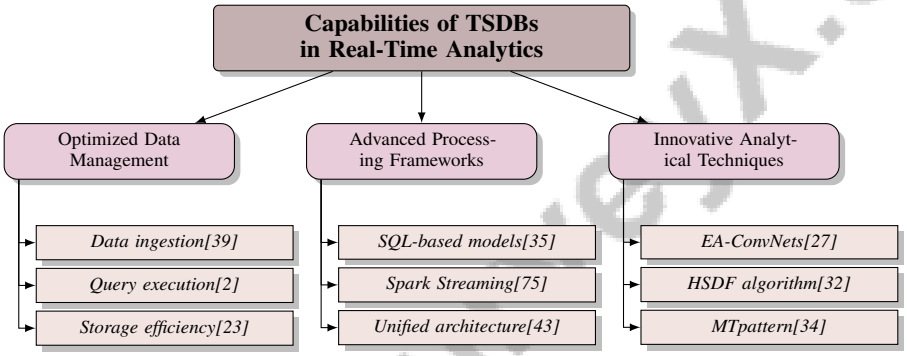

Figure 3: This figure illustrates the capabilities of TSDBs in real-time analytics by highlighting three primary categories: optimized data management, advanced processing frameworks, and innovative analytical techniques. Each category is supported by specific methodologies and technologies that enhance the performance and efficiency of TSDBs in handling real-time data streams.

4.2 Stream Processing Techniques in TSDBs

Stream processing techniques in TSDBs are vital for managing continuous data streams in real-time, enabling immediate insights and supporting dynamic decision-making. Unlike traditional batch processing systems, stream processing frameworks offer lower latency and better fault tolerance, ideal for high-velocity data environments [5, 76, 77, 41]. Recent advancements have improved the handling of large volumes of continuous data, facilitating interactive jobs and real-time queries.

A key aspect of stream processing in TSDBs is hierarchical aggregation of sensor data, as demonstrated by the Titan Stream Processing Architecture, which supports multiple hierarchies and runtime reconfiguration for scalable multidimensional aggregation [31]. Spark Streaming further showcases the power of real-time event processing, enabling applications like Car Information Systems to process data streams instantaneously, providing immediate feedback [75].

Techniques such as fixed-capacity rolling subsampling, including DStream, curate data streams by retaining only relevant data points, optimizing storage and processing resources [10]. Window Expressions enhance analysis and interpretation by facilitating precise segmentation of data streams [64].

To boost stream processing efficiency, runtime-optimized multi-way stream join methods continuously collect statistical data to dynamically adjust processing strategies, ensuring responsive and accurate real-time analytics [78]. Execution of SQL queries directly on streaming data, as enabled by systems like SSQM, expands the analytical capabilities of distributed stream processing systems, supporting complex operations on real-time data streams [38].

4.3 Frameworks and Algorithms for Real-Time Data Processing

Frameworks and algorithms for real-time data processing in TSDBs are essential for managing continuous data influx and ensuring timely insights. Platforms like Spark Streaming have demonstrated substantial performance improvements over traditional methods, particularly for real-time data processing applications [75]. The integration of heterogeneous signal processing systems, including CPUs, GPUs, FPGAs, and ASICs, enhances TSDB versatility and performance, particularly in sectors like IoT where large volumes of time-series data are generated. Optimized caching and filtering techniques ensure rapid data retrieval and processing, crucial for maintaining performance in dynamic scenarios [44, 2].

SQL-based streaming query models, implemented in frameworks like Apache Calcite, Flink, and Beam, have proven effective in real-world streaming use cases. These models integrate robust streaming capabilities into standard SQL interfaces, reducing barriers for developers and facilitating swift deployment of data processing solutions. This integration allows for comprehensive streaming analysis while leveraging familiar SQL constructs, enabling efficient management of high-velocity data streams [38, 79].

Advanced machine learning models, such as the Evolutionary State Graph Network (EvoNet), utilize graph neural networks to encode dynamic evolutionary state graphs, capturing changing relationships over time. This enhances predictions in time-series data by modeling both node-level and graph-level interactions, providing valuable insights into event occurrences [9, 80, 81, 14]. Innovative approaches, including vision transformers for learning cross-modality information using time-frequency spectrograms, significantly enhance forecasting accuracy, demonstrating over a 15% reduction in mean squared error compared to conventional models [82, 48, 18]. The integration of real-time storage and processing engines through shared memory architectures further enhances real-time analytics efficiency.

The Titan architecture exemplifies the aggregation of sensor data streams based on hierarchical groupings, essential for real-time analysis and reporting across various applications, including fraud detection and system monitoring. As traditional batch processing methods fall short of meeting immediate needs, advanced stream processing systems like AlertMix, Amazon Kinesis, and Apache Spark Streaming are increasingly adopted for continuous data analysis and critical business decision support [60, 41, 42, 43].

Frameworks and algorithms for real-time data processing in TSDBs are crucial for managing continuous data flows, especially in IoT and big data applications. These tools enable organizations to derive immediate insights from vast volumes of time-series data, facilitating dynamic decision-making across diverse sectors. Specialized Time Series Management Systems (TSMSs) have emerged to address traditional DBMS limitations, offering functionalities tailored for high-velocity data generation and real-time analysis. Benchmarking frameworks like IoTDB-Benchmark are vital for evaluating TSDB performance in real-world scenarios, ensuring businesses select the most suitable systems for their needs [23, 39, 79, 2, 44].

4.4 Real-Time Data Aggregation and Anomaly Detection

Real-time data aggregation and anomaly detection are crucial for managing time-series data streams, enabling meaningful insights and identifying abnormal patterns. Advanced aggregation techniques efficiently summarize data, facilitating timely decision-making and alleviating computational burdens associated with large data volumes. Techniques like adaptive time-series deep learning networks (ATSDLN) have demonstrated superior accuracy and adaptability in anomaly detection [83].

Anomaly detection is enhanced through non-parametric methods leveraging density and distance-based approaches, effectively identifying outliers in environments with unknown or variable data distributions [84]. Adjusting misclassification costs significantly improves detection accuracy, especially in scenarios where anomalies are rare yet critical [85].

Employing backtest overfitting measures, including combinatorially symmetrical cross-validation and probabilistic Sharpe ratios, offers a comprehensive framework for assessing anomaly detection algorithm performance, ensuring robustness and reliability over time [86].

Real-time data aggregation techniques, such as the Cloud Profiler (CP) method, exemplify the potential for significantly increasing data ingestion rates while maintaining high measurement

accuracy, achieving improvements from 700 k to 4.68 M tuples per second [87]. The integration of advanced aggregation and anomaly detection techniques is essential for managing real-time data streams, providing tools to extract valuable insights and identify critical anomalies promptly. Recent advancements in TSMSs and data augmentation techniques enhance efficiency, accuracy, and responsiveness, enabling effective handling of increasing real-time data volumes from diverse sources [23, 88, 66].

4.5 Scalability and Performance in Real-Time Stream Processing

Scalability and performance are critical in TSDBs due to the exponential data growth from applications like IoT, necessitating specialized TSDBs capable of handling significant data influx while providing rapid analytics. Benchmarking these databases with real datasets is essential to ensure performance metrics reflect actual conditions, as traditional benchmarks often utilize synthetic data that fails to represent real-world scenarios [60, 39, 2]. Efficient management and analysis of large-scale time-series data are crucial for businesses leveraging real-time insights across various domains.

Runtime-optimized multi-way stream join operators enhance processing efficiency, dynamically modifying strategies in response to real-time data characteristics, utilizing techniques like inter-operator feedback and progressive window widening to minimize computational overhead and improve response times [59, 61, 63]. Incremental query processing further enhances performance and reduces memory usage, facilitating real-time analytics on large datasets.

The Nephele/PACT streaming architecture significantly advances stream processing by addressing latency and throughput challenges. Its distributed scheme dynamically detects and optimizes for user-defined Quality of Service (QoS) constraints, reducing processing latency by at least a factor of 13 while ensuring high data throughput, particularly beneficial for large-scale applications requiring near-real-time handling of continuous data streams [61, 77]. This architecture highlights the importance of optimizing processing frameworks to mitigate latency challenges in real-time stream processing. Push-based streaming approaches have also shown competitive performance against traditional pull-based designs, achieving up to twice the performance while minimizing processing latency.

Selecting appropriate hardware is crucial for enhancing real-time stream processing performance, especially as modern applications increasingly favor Ethernet for data transport. This shift is driven by the need for efficient integration of ingestion, storage, and processing layers, significantly reducing processing latency and improving throughput. Recent advancements in streaming architectures illustrate the benefits of both push-based and pull-based mechanisms, facilitating better scalability and QoS management for large volumes of continuous data streams [61, 77]. This trend emphasizes the necessity of careful hardware configuration considerations to enhance processing capabilities and data transport efficiency.

The Titan architecture serves as a robust framework for scalable and reliable multidimensional aggregation of sensor data streams, effectively managing increasing data volumes from multiple sensors while demonstrating linear scalability as sensor counts grow. It ensures operational reliability amidst faults, facilitating continuous data processing and aggregation across hierarchical sensor groupings, which is essential for real-time analytics in applications like IoT and performance monitoring [35, 31, 60, 72, 43].

Despite advancements, challenges persist in achieving comprehensive fault tolerance and persistent state management, critical for ensuring the reliability of real-time stream processing systems in vital applications. The benchmark for Spark Streaming highlights its strengths, including in-memory processing, scalability, and fault tolerance, which are vital for real-time data analysis [75].

To effectively address scalability and performance challenges in real-time stream processing within TSDBs, a comprehensive strategy is essential. This strategy should incorporate advanced stream processing techniques from open-source frameworks like Apache Storm, Apache Spark, and Apache Flink, alongside commercial solutions like Amazon Kinesis and IBM Infosphere Stream. Additionally, optimizing hardware configurations and designing robust architectures are crucial for efficient data management, adaptable to the dynamic nature of data from diverse sources, enabling real-time analytics and processing while meeting the unique requirements of various application domains, including online gaming and IoT devices [76, 60, 38, 7, 41].

5 Data Aggregation Techniques

5.1 Overview of Data Aggregation Techniques

Benchmark	Size	Domain	Task Format	Metric
TreNet[14]	2,075,259	Time Series Analysis	Trend Prediction	RMSE
LTSM-bundle[89]	1,000,000	Time Series Forecasting	Forecasting	MSE, MAE
PLA-Bench[72]	1,500,000	Data Streaming	Data Compression	Compression Ratio, Latency
IoTDB-Benchmark[39]	3,000,000	Industrial Iot	Data Ingestion And Querying	Cost-time, Throughput
LLM4TS[90]	1,000,000	Time Series Analysis	Forecasting	MSE, MAE
Time-MMD[82]	1,000,000	Public Health	Time Series Forecasting	MSE, RMSE
COVID-VIS[65]	999	Epidemiology	Data Visualization Interpretation	Accuracy, Consensus
AIoT-Bench[70]	3,500,000	Time Series Analysis	Performance Evaluation	Runtime, Memory Usage

Table 4: This table presents a comprehensive overview of various benchmarks utilized in time-series data analysis, detailing their respective sizes, domains, task formats, and evaluation metrics. It highlights the diverse applications and methodologies employed in different benchmarks, ranging from trend prediction in industrial IoT to data visualization in epidemiology. The table serves as a valuable resource for understanding the scope and focus of each benchmark within the context of advancing data aggregation techniques.

Data aggregation techniques are crucial for efficiently managing and analyzing time-series data, facilitating the summarization of extensive datasets to derive actionable insights. These techniques enhance storage and computational efficiency in high-velocity environments by integrating ingestion, storage, and processing systems via advanced architectures employing both push and pull mechanisms. The exponential data growth challenges traditional algorithms in terms of resource demands and processing time. Techniques like sampling and probabilistic modeling reduce data size, enhance system throughput, and enable effective real-time analysis and decision-making [69, 61, 45]. The main aim of data aggregation is to simplify data complexity while preserving essential characteristics, thereby improving query and analytical efficiency.

Delta summation, a key aggregation method, focuses on differences between consecutive data points, significantly reducing computational demands in scenarios with incremental data changes [14]. Benchmark evaluations on various datasets, including household voltage and stock prices, demonstrate its adaptability and effectiveness. Table 4 provides a detailed comparison of benchmarks relevant to the discussion on data aggregation techniques, illustrating the diversity in domains, task formats, and metrics used to assess their performance.

Incorporating machine learning techniques, particularly recurrent neural networks (RNNs), into data aggregation enhances predictive capabilities in time-series analysis by leveraging temporal dependencies to improve prediction accuracy [91].

Statistical compression methods, such as the Elf method, enhance storage efficiency by increasing trailing zeros in floating-point values, yielding better compression ratios than conventional techniques [58]. These methods optimize storage while ensuring data integrity, enabling effective aggregation and retrieval.

Advanced augmentation strategies, including those by Demirel et al., enhance aggregation models' generalization capabilities, especially in contrastive learning tasks. By leveraging unlabeled data through pseudo-labeling, these strategies improve classification performance, facilitating more accurate analyses [91]. The Exathlon benchmark provides a framework for evaluating explainable anomaly detection in time-series data, incorporating both undisturbed and disturbed traces with ground truth labels for various anomaly types [92].

The integration of advanced stream processing techniques, as demonstrated by TiLT, achieves significant performance improvements in stream processing environments, enhancing throughput and efficiency. Selecting appropriate data cleaning techniques is critical for effective time-series data aggregation, ensuring that aggregated data remains accurate and actionable [50].

Data aggregation techniques in time-series analysis are vital for efficiently summarizing large datasets and extracting valuable insights. They facilitate dimensionality reduction, enhance machine learning training by reducing noise, and improve the interpretability of complex patterns. Methods like Symbolic Aggregate approXimation (SAX) and its modified versions effectively capture trends while

maintaining computational efficiency. The adaptive Brownian bridge-based aggregation (ABBA) and its variant, fABBA, significantly improve runtime and accuracy in processing time-series data. Additionally, advanced approaches like TOkenized Time Series EMbeddings (TOTEM) and novel spatiotemporal feature representations enhance the analysis of multidimensional time series, driving informed decision-making across various domains [73, 24, 12, 62, 26]. The continuous evolution of data aggregation strategies reflects their critical role in modern data analytics, addressing the complexities of time-series data and supporting dynamic decision-making processes.

5.2 Delta Summation and CRTS Methods

Delta summation is a specialized aggregation technique that enhances query execution times by utilizing differences between consecutive data points rather than absolute values, effectively reducing computational overhead and optimizing processing efficiency [93]. It is particularly beneficial in high-frequency data environments, such as multivariate time series datasets recording metrics like CPU time and network traffic at second-level granularity [94].

Benchmarking on an Intel Xeon E5-2650 v2 CPU demonstrates delta summation's efficiency across various data types and sizes, including datasets with up to 1 billion data points, highlighting its capability to process large-scale datasets efficiently [58]. The tsdownsample tool exemplifies delta summation's high performance and integration ease into existing visualization frameworks, effectively handling extensive datasets [58].

The Continuous Real-time Time Series (CRTS) method represents an emerging approach in time-series data aggregation, emphasizing continuity and real-time processing capabilities. It ensures aggregated outputs reflect the dynamic nature of underlying data streams, making it suitable for applications demanding real-time insights, such as continuous aggregation and analysis of large-scale data streams from social media platforms. Utilizing an efficient archival framework like the Tempo-Spatial Content Delivery Network (TS-CDN), which employs hash functions to minimize data redundancy and enhance unique data retrieval, organizations can effectively monitor and explore dynamic data across various spatial dimensions. The integration of push-based and pull-based streaming architectures further optimizes data ingestion and processing, significantly reducing latency and improving throughput for real-time analysis [95, 61, 42].

Combining delta summation and CRTS methods provides a robust framework for time-series data aggregation, addressing the challenges of efficient data processing and real-time analysis. By employing advanced techniques such as sampling for data reduction and distributed streaming SQL for real-time processing, data management systems can significantly enhance performance and scalability. These methods tackle the challenges posed by the exponential growth of big data—characterized by its volume, velocity, variety, and veracity—ensuring that systems remain agile and capable of efficiently meeting modern data demands. Furthermore, innovations in streaming architectures, including the integration of push-based and pull-based mechanisms, optimize data ingestion and processing, improving responsiveness and throughput in real-time analytics [69, 38, 61, 41].

5.3 Symbolic and Compression Techniques

Symbolic representation and compression techniques are pivotal in efficiently aggregating time-series data, transforming complex data into manageable forms while preserving essential information. These techniques are particularly beneficial for managing high-volume and high-dimensional data, facilitating efficient processing and analysis through methods like Symbolic Aggregate Approximation (SAX) for dimensionality reduction, probabilistic models for addressing data uncertainty in streaming contexts, and innovative approaches like Zoom-SVD for extracting latent patterns within arbitrary time ranges [69, 62, 11, 45].

The fABBA method exemplifies a symbolic representation approach that enhances computational efficiency and accuracy by employing a sorting-based strategy to aggregate time-series data [73]. This method transforms continuous time-series data into discrete symbols, allowing for more straightforward analysis and interpretation. By reducing data complexity, fABBA facilitates the identification of patterns and trends that may not be immediately apparent in the raw data.

Compression techniques complement symbolic representation by further reducing data size while maintaining critical information. These methods are vital for enhancing storage and retrieval pro-

cesses, particularly in resource-constrained environments or those experiencing high data throughput. Approaches integrating push-based and pull-based streaming architectures enable more efficient data ingestion and processing, reducing latency and increasing overall system throughput. Additionally, sampling techniques play a crucial role in managing large data volumes by effectively reducing data size while maintaining analytical accuracy, essential for timely data processing and decision-making in big data contexts. Frameworks like the Tempo-Spatial Content Delivery Network (TS-CDN) demonstrate how innovative data management strategies can significantly decrease storage requirements and improve data accessibility and analysis efficiency in large-scale social media datasets [69, 42, 61]. Advanced compression algorithms leveraging statistical properties of time-series data can achieve significant reductions in data volume without sacrificing accuracy.

The integration of symbolic and compression techniques in time-series data aggregation provides a robust framework for managing large datasets efficiently. By employing advanced data transformation and compression techniques, such as the TS-CDN and modified SAX, these methods enhance the efficiency of querying and analysis. The TS-CDN framework significantly reduces data redundancy through hashing, achieving a notable decrease in media file sizes while enabling effective monitoring of time-sensitive social media data. Meanwhile, the enhanced SAX technique captures segment trend information without sacrificing efficiency or query result accuracy. Together, these approaches facilitate timely decision-making and actionable insights across a wide range of applications, particularly in dynamic environments like social media and time series analysis [62, 42]. The continuous development of these techniques reflects their importance in addressing the complexities of modern data environments, ensuring that time-series data remains accessible and actionable.

5.4 Clustering and Spectrogram-Based Aggregation

Clustering and spectrogram-based aggregation techniques are pivotal in analyzing and managing time-series data, offering methods to effectively group and summarize complex datasets. Clustering techniques, such as k-means and hierarchical clustering, are widely used to identify natural groupings within time-series data, facilitating the discovery of patterns and trends that may not be immediately apparent. These methods are particularly useful in scenarios with high dimensionality and variability, allowing for the segmentation of data into meaningful clusters that reflect underlying structures [28].

Spectrogram-based methods enhance time-series data aggregation by transforming time-domain data into the frequency domain, providing a comprehensive view of the temporal and frequency characteristics of the data. This transformation is achieved through techniques like the Short-Time Fourier Transform (STFT), enabling the analysis of time-varying frequency content and capturing dynamic changes in the data over time [52]. By leveraging spectrograms, analysts can identify periodic patterns and anomalies indicative of significant events or trends, enhancing the interpretability and utility of the data.

As illustrated in Figure 4, which depicts the hierarchical classification of techniques used in clustering and spectrogram-based aggregation for time-series data analysis, the integration of these methodologies forms a comprehensive framework. This framework facilitates extracting significant insights from intricate datasets by effectively reducing dimensionality, capturing essential trends, and improving computational efficiency. Moreover, the versatility of methods like the matrix profile further enhances this framework, enabling efficient solutions for various time-series data mining tasks, including motif discovery and classification, across diverse fields such as bioinformatics and human activity monitoring [62, 73, 25]. These techniques are particularly valuable in applications like anomaly detection and trend analysis, where identifying patterns and changes in data is critical for timely decision-making. The continuous development of these methods underscores their importance in modern data analytics, ensuring that time-series data remains accessible and actionable across various domains.

5.5 Advanced Aggregation Techniques for Scalability

Advanced aggregation techniques are crucial for enhancing the scalability and performance of time-series data management systems, especially as data volumes continue to grow exponentially. The techniques discussed in the literature emphasize optimizing data processing and storage methods, such as sampling and stream processing, to improve the efficiency and effectiveness of deriving insights from large-scale datasets. These methods address the challenges posed by the exponential

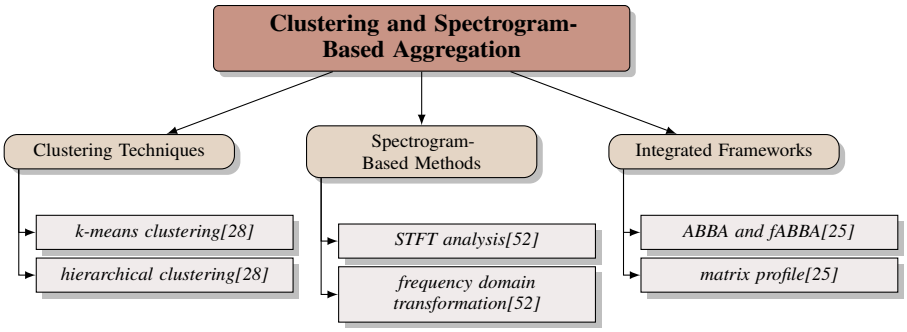

Figure 4: This figure illustrates the hierarchical classification of techniques used in clustering and spectrogram-based aggregation for time-series data analysis, highlighting the primary methods and their integration into comprehensive frameworks.

growth of data, particularly the high volume characteristic of big data, complicating traditional data mining and machine learning approaches. Strategies like structured downsampling and unified streaming architectures minimize resource demands and processing latency, enabling faster and more accurate analysis across diverse data types and applications [69, 10, 42, 61, 41].

One promising area of research involves developing more sophisticated downsampling algorithms that can be integrated into existing data management libraries. These algorithms aim to reduce data resolution while preserving critical information, thus optimizing storage and query performance. The tsdownsample tool exemplifies this approach by providing high-performance downsampling capabilities easily integrated into visualization frameworks [58]. Future research could focus on expanding this library to include a broader range of downsampling algorithms and enhancing its capabilities for distributed computing, thereby improving its scalability and applicability in diverse data environments.

In addition to downsampling, advanced symbolic representation techniques offer significant potential for scalability. By transforming continuous time-series data into discrete symbols, these methods reduce data complexity and facilitate efficient processing and analysis. The fABBA method employs a sorting-based strategy to achieve efficient aggregation, enabling the identification of patterns and trends within large datasets [73].

Furthermore, integrating machine learning models, such as recurrent neural networks (RNNs) and graph neural networks, into aggregation processes can enhance predictive capabilities and improve scalability. These models leverage temporal dependencies to provide accurate forecasts and insights, supporting dynamic decision-making across various applications [91].

Advanced aggregation techniques play a pivotal role in enhancing the scalability and performance of time-series data management systems. By focusing on optimizing data processing and storage, these techniques ensure that insights can be derived efficiently and effectively, supporting the growing demands of modern data environments. The ongoing advancement of analytical methods for large-scale time-series data underscores their vital role in navigating the intricate challenges posed by such data. Techniques like persistent homology enable the integration of domain-specific knowledge, enhancing the analysis of time series through tailored feature augmentation. Meanwhile, the introduction of the matrix profile offers a scalable and parameter-free solution for all-pairs similarity searches, facilitating a wide array of data mining tasks across diverse fields, from seismology to bioinformatics. Additionally, new methodologies for change-point analysis in the frequency domain provide robust estimation techniques that maintain accuracy even in the presence of extreme outliers. Collectively, these developments enable timely and precise analyses across various domains, significantly improving our ability to derive insights from complex time-series data [17, 96, 25].

6 Scalability in Time-Series Databases

As the need for efficient data management intensifies, scalability in time-series databases (TSDBs) becomes increasingly critical. This section delves into the scalability challenges faced by TSDBs as they cope with the burgeoning data volumes from real-time applications. Understanding these

challenges is vital for identifying strategies that enhance TSDB performance and adaptability. Figure 5 illustrates the hierarchical structure of scalability in TSDBs, categorizing the challenges, architectural strategies, and techniques necessary to enhance scalability. The figure highlights the complexity of algorithmic challenges, the adaptability of TSDBs, and the importance of efficient data management. Furthermore, it details architectural strategies such as micro-services and advanced clustering, along with techniques like advanced indexing and machine learning integration, which are essential for managing large-scale time-series data environments. The following subsection explores the specific scalability challenges encountered in TSDBs, emphasizing the complexities and limitations that demand innovative data management and analysis approaches.

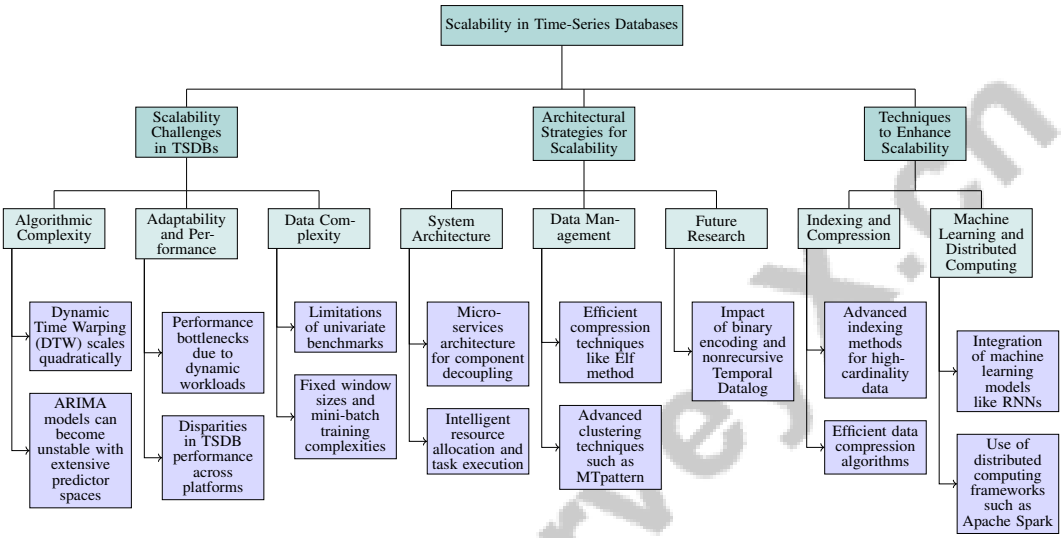

Figure 5: This figure illustrates the hierarchical structure of scalability in time-series databases (TSDBs), categorizing the challenges, architectural strategies, and techniques to enhance scalability. It highlights the complexity of algorithmic challenges, the adaptability of TSDBs, and the importance of efficient data management. The figure further details architectural strategies such as micro-services and advanced clustering, along with techniques like advanced indexing and machine learning integration, essential for managing large-scale time-series data environments.

6.1 Scalability Challenges in TSDBs

TSDBs face significant scalability challenges in handling the growing volumes of real-time data. A major issue is the computational complexity of algorithms like dynamic time warping (DTW), which scales quadratically, limiting its application in time-series clustering [56]. This complexity is further exacerbated by traditional models such as ARIMA, which can become unstable with extensive predictor spaces, highlighting the urgent need for scalable solutions within TSDBs [22].

Another critical challenge is the adaptability of TSDBs to dynamic workloads and fluctuating resource demands, often leading to performance bottlenecks in real-time data streams [43]. Jensen et al. emphasize the need for next-generation time-series management systems (TSMSs) capable of managing large-scale data while overcoming current performance and functionality limitations [23]. Benchmarking studies reveal disparities in TSDB performance, with systems like InfluxDB excelling in data loading and query execution, underscoring variability in scalability across platforms [2].

Many benchmarks focus on univariate time-series data, which do not fully capture the complexities of multivariate scenarios prevalent in real-world applications [19]. Models using fixed window sizes for local data features may face limitations, as optimal segment numbers vary across datasets, necessitating careful tuning. The computational complexity of methods like mini-batch training (O(T^2P))$further complicates scalability efforts in TSDBs$[20].

Despite these challenges, promising approaches exist. Xylogiannopoulos et al. propose methods that handle high-dimensional data while maintaining performance under noisy conditions [28]. However, current emphasis on time-series tables may not address general user queries or complex

causal reasoning, indicating areas for future development [30]. As TSDBs evolve, addressing these scalability challenges is crucial for managing modern data environments, providing timely and accurate insights across various applications.

6.2 Architectural Strategies for Scalability

Architectural strategies are essential for enhancing TSDB scalability, enabling efficient management of increasing data volumes and complex workloads. As illustrated in Figure 6, which highlights key architectural strategies for enhancing the scalability of Time Series Databases (TSDBs), the adoption of micro-services architecture, as demonstrated by systems like H-STREAM, integrates real-time data processing with historical analysis, ensuring scalability and adaptability through component decoupling [97]. This decoupling allows systems to dynamically adjust to varying demands, optimizing resource utilization.

Intelligent resource allocation and task execution management, exemplified by Nephele Streaming, balance high throughput and low latency by dynamically responding to workload changes [77]. Such adaptive resource management provides a robust framework for scalable TSDB operations, ensuring efficient processing and analysis of vast time-series data.

Efficient compression techniques, like the Elf method with O(N) time and O(1) space complexity, optimize storage and enhance query performance without compromising data integrity [98]. By reducing data footprints, these techniques enable TSDBs to manage larger datasets effectively, supporting scalability in storage and retrieval processes.

Advanced clustering techniques, such as MTpattern, address scalability challenges by minimizing cluster numbers while maintaining high accuracy [34]. This approach is beneficial in high-dimensional and complex time-series data environments, where efficient clustering is essential for scalable data management.

Future research includes exploring the impact of binary encoding on problem complexity and developing extensions of nonrecursive Temporal Datalog to ensure decidability for proposed decision problems [21]. These explorations could further enhance TSDB scalability by providing solutions to complexities associated with large-scale time-series data environments.

Architectural strategies enhancing TSDB scalability are crucial for effective data management, particularly as data from IoT and big data sectors grow exponentially. With numerous TSDBs available, each with unique architectures and performance characteristics, implementing benchmarks that reflect real-world data scenarios and advanced analytical operations is essential for organizations to select the most appropriate TSDB for their specific use cases [70, 39, 2, 38]. By leveraging micro-services, adaptive resource management, efficient compression, and advanced clustering techniques, TSDBs can maintain high performance and scalability, supporting dynamic decision-making across diverse applications.

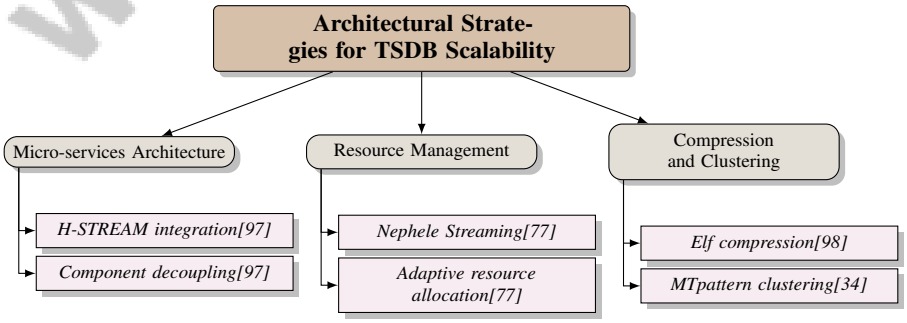

Figure 6: This figure illustrates key architectural strategies for enhancing the scalability of Time Series Databases (TSDBs), focusing on micro-services architecture, resource management, and compression and clustering techniques.

6.3 Techniques to Enhance Scalability

Enhancing scalability in TSDBs is crucial for managing the increasing volume and complexity of time-series data. Advanced indexing methods optimize query performance and resource utilization, with high-cardinality indexing enabling rapid data retrieval and analysis [39]. This approach is particularly beneficial in environments with high data velocity and volume, where quick access to specific data points is crucial.

Efficient data compression algorithms reduce data size while preserving essential information, optimizing storage and retrieval processes. Methods like the Elf method, which increases the number of trailing zeros in floating-point values, exemplify the potential for improving storage efficiency [98]. By minimizing data footprints, these techniques enable TSDBs to manage larger datasets effectively, supporting scalable data management.

Integrating machine learning models, such as recurrent neural networks (RNNs) and graph neural networks, enhances predictive capabilities and scalability. These models leverage temporal dependencies to provide accurate forecasts and insights, supporting dynamic decision-making across various applications [91]. Machine learning enables TSDBs to adapt to changing data patterns and demands, ensuring responsiveness and efficiency.

Distributed computing frameworks are pivotal in enhancing scalability by enabling parallel processing and resource sharing across multiple nodes. Systems like Apache Spark and Flink leverage distributed architectures to handle large-scale data processing tasks, ensuring TSDBs can efficiently manage and analyze vast amounts of time-series data [75]. These frameworks provide the infrastructure necessary for scalable data processing, allowing TSDBs to maintain high performance as data volumes grow.

To effectively enhance TSDB scalability, it is crucial to implement advanced techniques such as sophisticated indexing methods, optimized data compression algorithms, integration of machine learning for improved analysis, and adoption of distributed computing architectures. These strategies are particularly important for managing the substantial volumes of time-series data generated by IoT devices, addressing challenges in data storage, processing efficiency, and real-time querying capabilities [44, 69, 2]. By leveraging these techniques, TSDBs can effectively manage the complexities of large-scale time-series data environments, providing timely and accurate insights across diverse applications.

7 Conclusion

7.1 Future Directions and Research Opportunities

The landscape of time-series databases (TSDBs) and real-time analytics is poised for transformative growth, driven by innovative research avenues and technological progress. One promising area involves refining heterogeneous Probabilistic Finite State Automata (PFSA) structures, with a focus on extensive testing using diverse, real-world datasets. Expanding benchmarking efforts to include a broader array of datasets and additional TSDBs can significantly enhance the scope and relevance of performance assessments.

Efforts to improve the precision of time-series models might benefit from exploring alternative loss functions and incorporating longer symbolic subsequences, potentially boosting classification accuracy. Advances in discretization methods and the application of innovative techniques like 3CP to varied discrete sequences, including those in biological contexts, offer exciting research prospects.

Understanding the interplay between behavior patterns and external influences, as well as fortifying model resilience against sensor inaccuracies, are crucial areas for future exploration. The development of automated strategies for selecting Probabilistic Signal Temporal Logic (PSTL) templates, and extending these techniques to supervised and semi-supervised learning environments, presents substantial opportunities for progress.

The automation of segment selection in dynamic temporal pooling, with applications extending beyond traditional time series classification, represents another fertile research domain. Additionally, the creation of hybrid models that leverage the strengths of both statistical and machine learning methodologies could yield more robust predictive capabilities.

Advancing visualization and representation techniques in time-series data by expanding frameworks to support a wider variety of table types, enhancing user interaction, and scaling datasets for broader applicability is essential. Furthermore, optimizing transformation processes and investigating new types of transformations could greatly enhance the ability to detect similarities in time-series queries.

These research directions hold the potential to significantly advance the capabilities of time-series databases and real-time analytics, equipping these systems to meet the complex demands of diverse applications and rapidly evolving data landscapes.

References

[1] Shoaib Ahmed Siddiqui, Dominik Mercier, Mohsin Munir, Andreas Dengel, and Sheraz Ahmed. Tsviz: Demystification of deep learning models for time-series analysis, 2020.

[2] Bonil Shah, P. M. Jat, and Kalyan Sashidhar. Performance study of time series databases, 2022.

[3] Marcell Vazquez-Chanlatte, Jyotirmoy V. Deshmukh, Xiaoqing Jin, and Sanjit A. Seshia. Logic-based clustering and learning for time-series data, 2017.

[4] Jonas Van Der Donckt, Jeroen Van Der Donckt, Michael Rademaker, and Sofie Van Hoecke. Data point selection for line chart visualization: Methodological assessment and evidence-based guidelines, 2023.

[5] Marios Fragkoulis, Paris Carbone, Vasiliki Kalavri, and Asterios Katsifodimos. A survey on the evolution of stream processing systems, 2023.

[6] Roger S. Barga, Jonathan Goldstein, Mohamed Ali, and Mingsheng Hong. Consistent streaming through time: A vision for event stream processing, 2006.

[7] Álvaro Villalba and David Carrera. Multi-tenant pub/sub processing for real-time data streams, 2020.

[8] Fazl Barez, Paul Bilokon, and Ruijie Xiong. Benchmarking specialized databases for high-frequency data, 2023.

[9] Alexander Nikitin, Letizia Iannucci, and Samuel Kaski. Tsgm: A flexible framework for generative modeling of synthetic time series, 2024.

[10] Matthew Andres Moreno, Luis Zaman, and Emily Dolson. Structured downsampling for fast, memory-efficient curation of online data streams, 2024.

[11] Jun-Gi Jang, Dongjin Choi, Jinhong Jung, and U Kang. Zoom-svd: Fast and memory efficient method for extracting key patterns in an arbitrary time range, 2018.

[12] Sabera Talukder, Yisong Yue, and Georgia Gkioxari. Totem: Tokenized time series embeddings for general time series analysis, 2025.

[13] Davood Rafiei and Alberto Mendelzon. Similarity-based queries for time series data, 1998.

[14] Kouame Hermann Kouassi and Deshendran Moodley. An analysis of deep neural networks for predicting trends in time series data, 2020.

[15] Xiwen Chen, Peijie Qiu, Wenhui Zhu, Huayu Li, Hao Wang, Aristeidis Sotiras, Yalin Wang, and Abolfazl Razi. Timemil: Advancing multivariate time series classification via a time-aware multiple instance learning, 2024.

[16] Qianwen Meng, Hangwei Qian, Yong Liu, Yonghui Xu, Zhiqi Shen, and Lizhen Cui. Unsupervised representation learning for time series: A review, 2023.

[17] Eunwoo Heo and Jae-Hun Jung. Persistent homology of featured time series data and its applications, 2024.

[18] Panayiotis Christou, Shichu Chen, Xupeng Chen, and Parijat Dube. Test time learning for time series forecasting, 2024.

[19] Mohammad Braei and Sebastian Wagner. Anomaly detection in univariate time-series: A survey on the state-of-the-art, 2020.

[20] Brian Kenji Iwana. On mini-batch training with varying length time series, 2022.

[21] Alessandro Ronca, Mark Kaminski, Bernardo Cuenca Grau, Boris Motik, and Ian Horrocks. Stream reasoning in temporal datalog, 2018.

[22] Stephanie Clark, Rob J Hyndman, Dan Pagendam, and Louise M Ryan. Modern strategies for time series regression, 2020.

[23] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. Time series management systems: A survey, 2017.

[24] Xu Yan, Yaoting Jiang, Wenyi Liu, Didi Yi, and Jianjun Wei. Transforming multidimensional time series into interpretable event sequences for advanced data mining, 2024.

[25] Chin-Chia Michael Yeh. Towards a near universal time series data mining tool: Introducing the matrix profile, 2020.

[26] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. A review on outlier/anomaly detection in time series data, 2020.

[27] Wenlin Wang, Changyou Chen, Wenqi Wang, Piyush Rai, and Lawrence Carin. Earliness-aware deep convolutional networks for early time series classification, 2016.

[28] Konstantinos F. Xylogiannopoulos. Data curves clustering using common patterns detection, 2020.

[29] Dongha Lee, Seonghyeon Lee, and Hwanjo Yu. Learnable dynamic temporal pooling for time series classification, 2021.

[30] Jianing Hao, Zhuowen Liang, Chunting Li, Yuyu Luo, Jie Li, and Wei Zeng. Vistr: Visualizations as representations for time-series table reasoning, 2024.

[31] Sören Henning and Wilhelm Hasselbring. Scalable and reliable multi-dimensional aggregation of sensor data streams, 2019.

[32] Adedotun Akintayo and Soumik Sarkar. Hierarchical symbolic dynamic filtering of streaming non-stationary time series data, 2017.

[33] Etienne Le Naour, Ghislain Agoua, Nicolas Baskiotis, and Vincent Guigue. Interpretable time series neural representation for classification purposes, 2023.

[34] Rohan Kabra, Divya Saxena, Dhaval Patel, and Jiannong Cao. Time series clustering for human behavior pattern mining, 2021.

[35] Janak Dahal, Elias Ioup, Shaikh Arifuzzaman, and Mahdi Abdelguerfi. Distributed streaming analytics on large-scale oceanographic data using apache spark, 2019.

[36] Shuai Zhang, Wenxi Zeng, I-Ling Yen, and Farokh B. Bastani. Semantically enhanced time series databases in iot-edge-cloud infrastructure, 2019.

[37] Chin-Chia Michael Yeh, Yan Zheng, Junpeng Wang, Huiyuan Chen, Zhongfang Zhuang, Wei Zhang, and Eamonn Keogh. Error-bounded approximate time series joins using compact dictionary representations of time series, 2023.

[38] Milinda Pathirage and Beth Plale. Fast data management with distributed streaming sql, 2015.

[39] Rui Liu, Jun Yuan, and Xiangdong Huang. Benchmarking time series databases with iotdb-benchmark for iot scenarios, 2024.

[40] Ahmed Shifaz, Charlotte Pelletier, Francois Petitjean, and Geoffrey I. Webb. Ts-chief: A scalable and accurate forest algorithm for time series classification, 2020.

[41] Saeed Shahrivari and Saeed Jalili. Beyond batch processing: Towards real-time and streaming big data, 2014.

[42] Melika Bahman-Abadi and M. B. Ghaznavi-Ghoushchi. Analysis and extraction of tempo-spatial events in an efficient archival cdn with emphasis on telegram, 2023.

[43] Ayush Singhal, Rakesh Pant, and Pradeep Sinha. Alertmix: A big data platform for multi-source streaming data, 2018.

[44] R. S. Weigel, D. M. Lindholm, A. Wilson, and J. Faden. Tsds: high-performance merge, subset, and filter software for time series-like data, 2010.

[45] Yanlei Diao, Boduo Li, Anna Liu, Liping Peng, Charles Sutton, Thanh Tran, and Michael Zink. Capturing data uncertainty in high-volume stream processing, 2009.

[46] Gaurangi Anand and Richi Nayak. Unsupervised visual time-series representation learning and clustering, 2021.

[47] Jeroen Van Der Donckt, Jonas Van Der Donckt, Michael Rademaker, and Sofie Van Hoecke. Minmaxlttb: Leveraging minmax-preselection to scale lttb, 2023.

[48] Zhen Zeng, Rachneet Kaur, Suchetha Siddagangappa, Tucker Balch, and Manuela Veloso. From pixels to predictions: Spectrogram and vision transformer for better time series forecasting, 2024.

[49] Kamil Faber, Roberto Corizzo, Bartlomiej Sniezynski, Michael Baron, and Nathalie Japkowicz. Watch: Wasserstein change point detection for high-dimensional time series data, 2022.

[50] Matthew B. Kennel. Statistical test for dynamical nonstationarity in observed time-series data, 1995.

[51] Shuang Song and Kamalika Chaudhuri. Composition properties of inferential privacy for time-series data, 2017.

[52] Hyunseung Chung, Sumin Jo, Yeonsu Kwon, and Edward Choi. Time is not enough: Time-frequency based explanation for time-series black-box models, 2024.

[53] Vanessa Freitas Silva, Maria Eduarda Silva, Pedro Ribeiro, and Fernando Silva. Mhvg2mts: Multilayer horizontal visibility graphs for multivariate time series analysis, 2023.

[54] Michael Neuder, Elizabeth Bradley, Edward Dlugokencky, James W. C. White, and Joshua Garland. Detection of local mixing in time-series data using permutation entropy, 2020.

[55] Ricards Marcinkevics, Steven Kelk, Carlo Galuzzi, and Berthold Stegemann. Discovery of important subsequences in electrocardiogram beats using the nearest neighbour algorithm, 2019.

[56] Volkan Kumtepeli, Rebecca Perriment, and David A. Howey. Fast dynamic time warping and clustering in c++, 2023.

[57] Padraig Davidson, Michael Steininger, André Huhn, Anna Krause, and Andreas Hotho. Semi-unsupervised learning for time series classification, 2022.

[58] Jeroen Van Der Donckt, Jonas Van Der Donckt, and Sofie Van Hoecke. tsdownsample: high-performance time series downsampling for scalable visualization, 2023.

[59] Rafael Fernández-Moctezuma, Kristin Tufte, and Jin Li. Inter-operator feedback in data stream management systems via punctuation, 2009.

[60] Maninder Pal Singh, Mohammad A. Hoque, and Sasu Tarkoma. A survey of systems for massive stream analytics, 2016.

[61] Ovidiu-Cristian Marcu and Pascal Bouvry. Colocating real-time storage and processing: An analysis of pull-based versus push-based streaming, 2022.

[62] Muhammad Marwan Muhammad Fuad. Modifying the symbolic aggregate approximation method to capture segment trend information, 2020.

[63] David Tolpin. Progressive temporal window widening, 2017.

[64] M. Praveen and S. Hitarth. Window expressions for stream data processing, 2024.

[65] Oded Stein, Alec Jacobson, and Fanny Chevalier. The effect of smoothing on the interpretation of time series data: A covid-19 case study, 2023.

[66] Kexin Rong and Peter Bailis. Asap: Prioritizing attention via time series smoothing, 2017.

[67] Xi Wang and Chen Wang. Time series data cleaning with regular and irregular time intervals, 2020.

[68] Renjie Wu and Eamonn J. Keogh. Fastdtw is approximate and generally slower than the algorithm it approximates, 2022.

[69] Zhicheng Liu and Aoqian Zhang. A survey on sampling and profiling over big data (technical report), 2020.

[70] Prabhav Arora. A demonstration of benchmarking time series management systems in the cloud, 2021.

[71] Lukas Burkhalter, Anwar Hithnawi, Alexander Viand, Hossein Shafagh, and Sylvia Ratnasamy. Timecrypt: Encrypted data stream processing at scale with cryptographic access control, 2020.

[72] Romaric Duvignau, Vincenzo Gulisano, Marina Papatriantafilou, and Vladimir Savic. Piecewise linear approximation in data streaming: Algorithmic implementations and experimental analysis, 2018.

[73] Xinye Chen and Stefan Güttel. An efficient aggregation method for the symbolic representation of temporal data, 2022.

[74] Vidhi Agrawal, Gajraj Kuldeep, and Dhananjoy Dey. Near lossless time series data compression methods using statistics and deviation, 2022.

[75] Philipp M. Grulich. Scalable real-time processing with spark streaming: implementation and design of a car information system, 2017.

[76] Jeyhun Karimov, Tilmann Rabl, Asterios Katsifodimos, Roman Samarev, Henri Heiskanen, and Volker Markl. Benchmarking distributed stream data processing systems, 2019.

[77] Björn Lohrmann, Daniel Warneke, and Odej Kao. Nephele streaming: Stream processing under qos constraints at scale, 2013.

[78] Jinlong Hu and Tingfeng Qiu. Runtime-optimized multi-way stream join operator for large-scale streaming data, 2024.

[79] Edmon Begoli, Tyler Akidau, Fabian Hueske, Julian Hyde, Kathryn Knight, and Kenneth Knowles. One sql to rule them all, 2019.

[80] Wenjie Hu, Yang Yang, Ziqiang Cheng, Carl Yang, and Xiang Ren. Time-series event prediction with evolutionary state graph, 2020.

[81] Christian Wiwie, Alexander Rauch, Anders Haakonsson, Inigo Barrio-Hernandez, Blagoy Blagoev, Susanne Mandrup, Richard Röttger, and Jan Baumbach. Elucidation of time-dependent systems biology cell response patterns with time course network enrichment, 2017.

[82] Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Lingkai Kong, Harshavardhan Kamarthi, Aditya B. Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, and B. Aditya Prakash. Time-mmd: Multi-domain multimodal dataset for time series analysis, 2025.

[83] Hui Ye, Xiaopeng Ma, Qingfeng Pan, Huaqiang Fang, Hang Xiang, and Tongzhen Shao. An adaptive approach for anomaly detector selection and fine-tuning in time series, 2019.

[84] Evgeny Burnaev and Vladislav Ishimtsev. Conformalized density- and distance-based anomaly detection in time-series data, 2016.

[85] Tochukwu John Anih, Chika Amadi Bede, and Chima Festus Umeokpala. Detection of anomalies in a time series data using influxdb and python, 2020.

[86] Joel da Costa and Tim Gebbie. Learning low-frequency temporal patterns for quantitative trading, 2020.

[87] Shinhyung Yang, Jiun Jeong, Bernhard Scholz, and Bernd Burgstaller. Cloudprofiler: Tsc-based inter-node profiling and high-throughput data ingestion for cloud streaming workloads, 2023.

[88] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey, 2022.

[89] Yu-Neng Chuang, Songchen Li, Jiayi Yuan, Guanchu Wang, Kwei-Herng Lai, Leisheng Yu, Sirui Ding, Chia-Yuan Chang, Qiaoyu Tan, Daochen Zha, and Xia Hu. Understanding different design choices in training large time series models, 2024.

[90] Liangwei Nathan Zheng, Chang George Dong, Wei Emma Zhang, Lin Yue, Miao Xu, Olaf Maennel, and Weitong Chen. Revisited large language model for time series analysis through modality alignment, 2024.

[91] Berken Utku Demirel and Christian Holz. Finding order in chaos: A novel data augmentation method for time series in contrastive learning, 2023.

[92] Dominique Mercier, Andreas Dengel, and Sheraz Ahmed. Patchx: Explaining deep models by intelligible pattern patches for time-series classification, 2021.

[93] Derek Colley and Md Asaduzzaman. An improved method of delta summation for faster current value selection across filtered subsets of interval and temporal relational data, 2022.

[94] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul. Exathlon: A benchmark for explainable anomaly detection over time series, 2021.

[95] Daniel Loureiro, Kiamehr Rezaee, Talayeh Riahi, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. Tweet insights: A visualization platform to extract temporal insights from twitter, 2023.

[96] Gyorgy H. Terdik, Stergios B. Fotopoulos, and Venkata K. Jandhyala. Change-point analysis in frequency domain for chronological data, 2016.

[97] Genoveva Vargas-Solar and Javier A. Espinosa-Oviedo. Building analytics pipelines for querying big streams and data histories with h-stream, 2021.

[98] Ruiyuan Li, Zheng Li, Yi Wu, Chao Chen, Songtao Guo, Ming Zhang, and Yu Zheng. Erasing-based lossless compression method for streaming floating-point time series, 2023.

Disclaimer:

SurveyX is an AI-powered system designed to automate the generation of surveys. While it aims to produce high-quality, coherent, and comprehensive surveys with accurate citations, the final output is derived from the AI's synthesis of pre-processed materials, which may contain limitations or inaccuracies. As such, the generated content should not be used for academic publication or formal submissions and must be independently reviewed and verified. The developers of SurveyX do not assume responsibility for any errors or consequences arising from the use of the generated surveys.