Technische Universität Berlin

Chair of Database Systems and Information Management

Master's Thesis

Anonymized Access Control for Distributed Event Stores

Henri Tyl Allgöwer
Degree Program: Computer Science
Matriculation Number: 454925

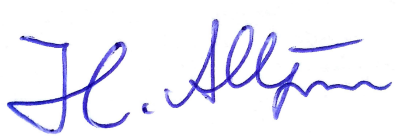
Reviewers
Prof. Dr. Volker Markl
Prof. Dr. Odej Kao

Advisor
Rudi Poepsel Lemaitre

Submission Date
30.11.2023

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, October 15, 2023



Henri Tyl Allgöwer

Zusammenfassung

Tipps zum Schreiben dieses Abschnitts finden Sie unter [8]

Abstract

The abstract should be 1-2 paragraphs. It should include:

- a statement about the problem that was addressed in the thesis,
- a specification of the solution approach taken,
- a summary of the key findings.

For additional recommendations see [8].

Acknowledgments

For recommendations on writing your Acknowledgments see [9]. Thank you to the chair at Database Systems and Information Management (DIMA)

Contents

Chapters 3-5 are the core of the thesis, whereas Chapters 1, 2, 6, and 7 provide context. The major contributions should be in Chapters 4 and 5. This structure serves as a guideline and should be customized accordingly. In particular, the generic chapter titles should be replaced with more specific ones, where appropriate (e.g., Chapter 4).

List of Figures

List of Tables

List of Abbreviations

DIMA Database Systems and Information Management

CCPA California Consumer Privacy Act

GDPR General Data Protection Regulation

ICD International Statistical Classification of Diseases and Related Health Problems

List of Algorithms

1 Introduction

... should include the following:

- motivation (why is this problem interesting? offer examples),
- research challenge (what is the obstacle to be overcome?),
- novelty (was this problem already solved?),
- anticipated impact (how does solving this problem impact our world?).

1.1 Motivation

The increasing popularity of data streaming in corporations highlights the imperative need for incorporating anonymization and data masking techniques in this technology. Particularly noteworthy is the extensive adoption of distributed event stores, which are scalable and fault-tolerant systems designed to capture, store, and process real-time data streams, across various sectors, including Fortune 100 companies, governments, healthcare, and transportation industries [5]. This widespread usage emphasizes the criticality of ensuring data privacy within these distributed event stores, while also highlighting the potential transformative impact of effective anonymization and data masking techniques in this domain.

1.2 Research Challenge

In the contemporary data-driven world, the growing demand for comprehensive data privacy policies is matched by increasingly stringent regulations from governments worldwide [1,2]. However, the underlying infrastructure to adequately support these policies is markedly lacking [3,4]. This disparity poses a unique challenge, particularly when considering the demands of modern database systems to maintain high performance, characterized by low latency and high throughput.

1.3 Novelty

While there exists a body of work focusing on anonymization and data masking for data streaming [6,7,8], there is a noticeable gap of research specifically targeting distributed event stores. Furthermore, although there are enterprise technologies for managing data flowing into such systems [9], there is limited literature on techniques designed for data already within. Most notably, the concept of integrating Role-Based Access Control (RBAC) within this framework, where the role assigned determines the level of anonymity accorded to the data, is a completely novel approach.

1.4 Anticipated Impact

The integration of data privacy policies with modern data stream structures, such as distributed event stores, would be a significant innovation. The introduction of Role-Based Access Control (RBAC) coupled with anonymization in distributed event stores holds the potential to contribute to more advanced, efficient, and secure data handling. By making such tools accessible and cost-effective, companies might be more inclined to prioritize and invest in user data privacy.

1.5 Research Problem

... should include the following:

- a succinct, precise, and unambiguous statement of the research problem or question to be solved,
- goals and subproblems that will be explored, including the scope of the thesis (i.e., what is in and out of scope).

Area (Million sq. miles)	Calling Code
0.29	56
0.3	90
3.8	1
0.5	51
600	9800
Pearson = 1.0	Spearman's = 0.1

Table 1: Correlation in the existence of outlier[5].

Is there also indentation here?
Hard to tell without two lines
Oh wow there is

1.6 Outline

2 Literature Review

2.1 Role Based Access Control

2.2 Distributed Event Stores

2.3 Anonymization

Techniques Cryptography, Masking Functions and there is a thirsd ?

2.3.1 Principles

2.3.2 Data Streaming

One prominent example of applying anonymization techniques to streamed data is the work CASTLE [1]

2.4 Related Work

... should include the following:

- definitions / technical terms,
- theoretical foundations / principles,
- descriptions of algorithms, hardware, software, and/or systems employed.

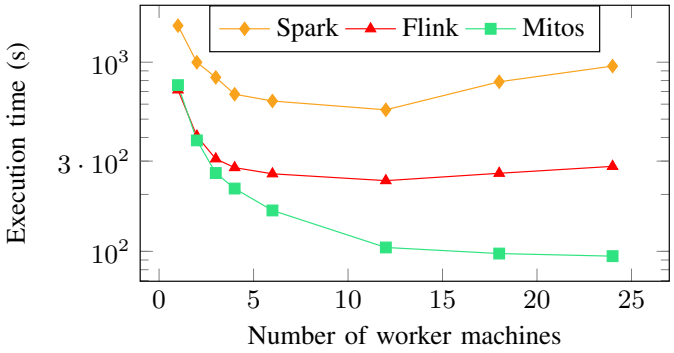


Figure 1: Strong scaling for Visit Count[3].

3 Theoretical Framework

3.1 Managing Different Anonymization Granularity

When planning to integrate anonymization techniques into existing systems, there are many things to consider. First one must understand the data flowing through the system. Does it include personally identifiable information? Is there further sensitive data? What part of it is necessary for system maintenance? Maybe there is additional data collected for statistics. Bearing this in mind the next thought would what needs to be anonymized. For this privacy agreements with the user must be taken into account. There may be additional government regulations in place like the California Consumer Privacy Act (CCPA) in the United States of America or the General Data Protection Regulation (GDPR) in the European Union. The next course of action is deciding on specific masking functions. As was discussed in 2.3 there is a plethora to chose from. It is important to note that all forms of anonymization lead to a loss of information. While choosing Blurring or Suppression, two methods that replace attributes with a placeholder, for all critical data fields derived in the previous assessment will ensure that all privacy concerns are addressed, it will also diminish all intelligence gained from collecting this data in the first place. It is questionable if not collecting this type of data in the first place would then be the better solution as it would save storage and computing power. An alternative approach would be to invest heavily in the IT security ensuring that no intruder with malicious intent can gain access to sensitive data. Keeping in mind, however, that social engineering attacks are nowadays the most common and effective strategy (SAUCE) giving a guarantee of safety can be impossible. It also stands to reason that the more employees a company has the risk for social engineering attacks increases. Both of these radical approaches do not seem to adequately solve the problem. Fortunately, there is a way to navigate between these two extremes. An attentive observer of the company's data operations will likely notice that data can and should be restricted similarily to permissions: Only the data needed to furfill the user's duty should be accessible to the user. In addition, there are anonymization techniques, which do not lead to total information loss like the two mentioned before. Generalization for example can be employed to significantly reduce the re-identification of individuals, while simultaneously retaining some information. With data restriction and

different anonymization techniques in mind there is a middle ground to be found which maximizes security and minimizes information loss. Consider the following example:

3.1.1 Use Case Example

Hospitals depend on the collection, management and analysis of data to administer the best and most accurate care of their patients. In a modern hospital all data would be stored in a centralized hospital database. Here all data for individual patients are brought together. Table 2 shows an exemplary entry of a patient in the diabetes and endocrinology ward of a german hospital. Note that the table has been shortened to enhance its clarity and readability. (A MORE DETAILED VERSION CAN BE FOUND IN THE APPENDIX)

pid	name	addr.	gender	age	ins. co.	ins. no.	diag.	gluc.	hba1c	med.
1	F. Ott	Berlin	M	28	TK	K15489	E10	22.1	8.74	Insulin

Table 2: Example datum of a diabetes patient

The header of table 2 shows eleven attributes. First the person id (*pid*), which is the primary key for each patient as it uniquely identifies the patient in the database. Then *name*, address (*addr.*), *gender* and *age* are included as additional personal information. This would typically also include the full address not just the city name, contact information, height as well as weight to adapt the dosage of the medication. The subsequent two attributes include information about the patient's insurance information. Insurance companies in Germany are uniquely identified with a nine digit institutional identifier. In this example the insurance company (*ins. co.*) has the value 101575519, which matches the identifier of the Techniker Krankenkasse (TK). Each client is then assigned a number unique to that insurance company called the insurance number (*ins. no.*). It always starts with a letter followed by digits. Finally, the datum references the medical information. It starts with the diagnosis (*diag.*) classified according to the International Statistical Classification of Diseases and Related Health Problems (ICD). E10 being the label for Type 1 Diabetes Mellitus. The most important medical measurement for the treatment of this disease is the current amount of glucose (*gluc.*) in the blood. This determines the quantity of medication (*med.*) to be administered to the patient. For Type 1 Diabetes this is Insulin. Lastly, the table includes an attribute called *hba1c*. This is the body's own three-month average of blood

glucose. By means of which diabetes is diagnosed. In this case it is also symbolic for all additional diagnostic findings. Glucose and HbA1c are intentionally distinguished as separate attributes in this dataset, despite both being blood-derived metrics, due to their distinct measurement methodologies and relevance in immediate treatment contexts. Glucose can be ascertained with a single drop of blood, providing critical information for the immediate treatment. Conversely, HbA1c is derived from a complete blood count and does not require instant action.

In a hospital setting, numerous actors engage with the aforementioned dataset. The most straightforward and prominent is the doctor. She will need all data to fulfill her duties. The doctor's letter contains all personal information. The medical data is needed for diagnosis and treatment. She will also need to keep the insurance information in mind as the covered treatment options are oftentimes different for each company. Additionally, she will need to write the patient's insurance information on the rescriptions. Only the pid could be omitted, but is debatable if the overhead is worth it, considering the pid can be easily inferred with all the given information. Therefore, no anonymization to the doctor's data makes the most sense.
Supporting the doctor is the nurse staff. One of their main tasks is to monitor patients and administer medication. To accomplish this they require the diagnosis, medication and in this case the glucose data. As the HbA1c value is not relevant for the immediate treatment it can be safely ommitted. Again insurance information is necessary as nurses typically do have the liberty of administer medication according to their own judgement. This is especially important when considering how understaffed hospitals in Germany are most of the time. On the other hand the patient's personal insurance number does not play into this. As nurses also interact directly with the patients they need some basic personal information like name and gender. Pid and address, however, are not required. Therefore, the data for the nurse staff can be anonymized as shown in Table 3 without limiting the nurses or losing valuable information.

pid	name	addr.	gender	age	ins. co.	ins. no.	diag.	gluc.	hba1c	med.
*	F. Ott	*	M	28	TK	*	E10	22.1	*	Insulin

Table 3: Data available for the nurse staff. Note that pid, address, insurance number and additional medical information.

In tandem with the stay and medical treatment of the patient, the administartion of the hospital will want to collect the money from the patient's insurance.

The insurance company together with the patient's personal insurance number will suffice as identification. Administered medication will be imperative as this dictates the amount of money the hospital will get in addition to the fees for the stay. For this the diagnosis will typically have to be added as a suitable reason. No further information is required. Limiting the amount of data here is crucial as here the data is exported to a third party. Which means that additional regulations will take effect. Minimizing the data leaving the hospital minimizes security risks. With these strict rules in place the data can be adjusted as seen in Table 4.
Note at this point that an attacker, who has gained access to both the data of the nurse staff and that of the administration, would struggle to correlate the entries. The shared available data fields insurance company, diagnosis and medication are likely generic enough to not point to a singular but to many patients.

pid	name	addr.	gender	age	ins. co.	ins. no.	diag.	gluc.	hba1c	med.
*	*	*	*	*	TK	K15489	E10	*	*	Insulin

Table 4: Data available for the administration. Note that only the insurance information, medication and diagnosis are not suppressed.

Diabetes, which afflicts over ten percent of the global population and demonstrates a rising prevalence, stands as one of the most common chronic diseases worldwide [2, 6]. Given its mostly non-lethal progression and lifetime dependency on medication, it has given rise to a substatial market. As cause, optimal treatment and cure remain subject to research, data of especially newer diabetes patients is in hot demand. To provide this data to research institutes in accordance with the regulations in place the hospital must ensure that no concrete patient can be reidentified. Here, advanced anonymization techniques such as K-Anonymization come into place. Each attribute of the data entry can be assigned to one of three categories: personally identifiable, quasi indentifying and remaining attributes. To achieve k anonymity each entry must supress the personally identifiable attributes, while keeping the remaining attributes untouched. Most importantly the quasi identifiable attributes of each data entry must be the same for at least k - 1 other entries of a data set. This is typically achieving with generalization of these attributes until k entries are found. In this use case the personally identifiable attributes are *pid*, *name*, *address* and *insurance number*. The quasi indentifying attributes are *gender*, *age* and *insurance company*. The medical data comprive the remaining attributes. A K anonymous version of this data extry is depicted in Table 5.

pid	name	addr.	gender	age	ins. co.	ins. no.	diag.	gluc.	hba1c	med.
*	*	*	{M, F, X}	[20 - 30]	ins. co.	*	E10	22.1	8.74	Insulin

Table 5: Data available for external research. Note the untouched medical, the suppressed personally identifying and generalized quasi identifiable attributes.

While the aforementioned diabetes patient use case scenario may appear unique and specific, the aspects and nuances are applicable in numerous contexts. The distinct data requirements for doctors, nurses, administration and research are anticipated to persist, albeit adapted, throughout the entire healthcare industry. It is also viable in different sectors. Imagine security levels in government matters, trade secrets and specific customer knowledge in corporations or secrecy of correspondense for the transportation industry. Distributed event stores are utilized across all of these sectors with major players relying on Apache Kafka for their day to day needs [4].

3.2 Masking Functions

Data anonymization is a multifaceted process tailored to meet application requirements, user groups and their specific needs. This section goes in depth into masking functions, the cornerstone of data anonymization. In this context a *datum* refers to a single piece of information e.g. a singular entry of a database or any one event of a data stream. Synonymous with it also used the word *tuple*. A tuple is comprived of one or multiple *attributes* (also called *fields*). It is customary for individual tuples to be part of a bigger collection. In the static context they are collected in databases and grouped in tables. Data streams work analogous for dynamic operations. All tuples of the same database table or data stream are required to follow the same pattern. This refers to the sequence of attributes of each tuple. This pattern is fixed in a data schema associated with the table or stream. Sometimes it is appended to each datum in form of a header as seen exemplary in Table 2. As this substantially increases the size of each datum it is more common to define it once in the initialization step of the database table or data stream.
While all forms of anonymization aim to alter the underlying sensitive information, e.g. the tuple's attributes containing perhaps personally identifiying information, in a way that makes it harder if not impossible to reconstruct, they can be categorized based on their scope of operation:

- **Value-Based** Handles one tuple at a time and replaces the values of at-

tributes independently.

- **Tuple-Based** Operates on individual attributes of a single tuple, but considers the values of the entire tuple for the change.
- **Attribute-Based** Extends the view from one tuple to a larger collection or table of data. Evaluates the values of singular attributes of the entire set and collectively makes changes to that attribute accordingly.
- **Table-Based** Covers a table of data and perceives all attributes of each tuple. Adaptions to multiple attributes simultaneously are common. It can be argued that methods falling under this category are not masking functions but algorithms utilizing many masking functions to achieve anonymization on a table level.

To better illuminate the relationship and hierarchy of the aforementioned categories as well as provide some examples, refer to Figure 2.

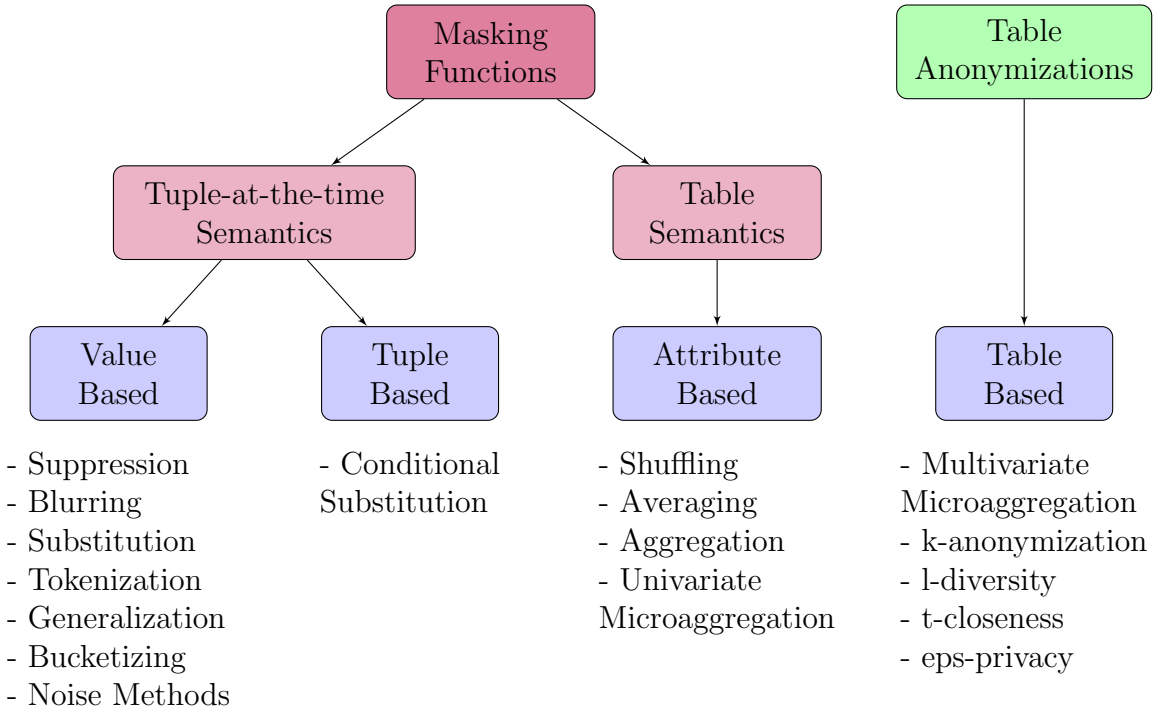


Figure 2: Hierarchy of masking functions

Having outlined the structure and dimensions of the various masking functions, it is now time to take a closer look at the functionality and use cases of the examples given in Figure 2 starting with the value based.

Suppression aims to effectively delete the value by replacing the value of an attribute with a meaningless character, most commonly the asterisk *. It is important to note, that actually removing the attribute from the tuple or replacing it will a null value would violate the data schema and thus negatively impact operability. The asterisk does the trick while maintaining the data schema. To work Suppression only requires a non-empty set of keys for the attributes that are supposed to be suppressed as parameter. Naturally, it leads to total information loss of the specfified fields. This can be particular useful for containing personally identifiable data like a person's home address as exemplary shown in Figure 3.

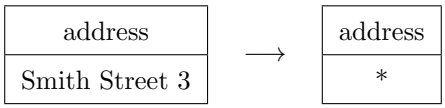


Figure 3: Example of suppression of an attribute.

A similar approach is applied in **Blurring**. Here the value of an attribute is replaced with arbitrary characters, typically Xs. It is distinguishable from Suppression in that not all characters of the value have to be replaced and even the amount of characters can remain the same. Imagine a user at the checkout of an online store that they have already purchased goods at prior to this session. Here the credit card information of that user was saved as part of the agreement from the previous session. The user is then given the option to use that credit card again, with it being specified as a sequence of blurred characters with only the last three digits in plain text as shown in Figure 4. This allows the user to double-check the card information without exposing the credit card number to the network, screen capturers or bystanders. This operation also leads to high information loss but retains some usability of the value. The parameters for Blurring include the keys to blurr as well as optionally the number of characters and whether the amount of characters is to be maintained. Note, that setting the parameters to blur all characters and reduce the amount to one is equal in funtionality to Suppression.

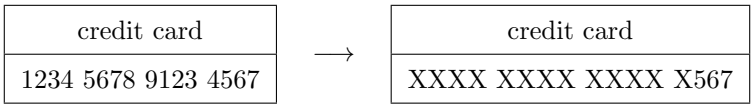


Figure 4: Example of blurring of an attribute.

Substitution replaces the value of the specified attribute with a predefined substitute. Figure 5 shows an example where a name is switched out with an

arbitrary fake name from a library. While this masking function leads to substantial information loss, it seemingly maintains the integrity of the data from an outsider's perspective. This can make the data easier to work with, while still ensuring anonymity. As parameters the keys for the attributes that are supposed to be substituted are required in tandem with the intended substitutes.

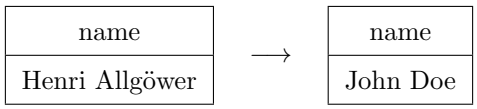


Figure 5: Example of substitution of an attribute.

An alternative approach is **Tokenization**. Here values are also substituted, but not with some arbitrary replacement. Instead, the substitute is a specific token. These tokens can be reversed to restore the original value as long as a secret is known with which the token was created. There are different approaches to achieve this. The first one coming to mind is a database mapping token to the hash of the original value. Only access to the database as well as the hashing function will yield the correct original value from the token. Another approach is to omit the database and instead use a more sophisticated cryptographic algorithm to create the token, essentially encrypting the data. While this masking function manages to preserve the information, it requires substantial overhead in form of storage for the database and/or computing power for the cryptographic algorithms. This costly masking function is usually reserved for highly sensitive data. For instance passwords as illustrated in Figure 6.

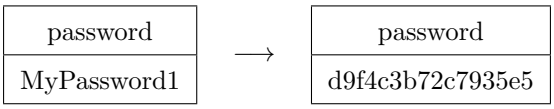


Figure 6: Example of tokenization of an attribute.

One of the most common masking functions is **Generalization**. Here, the value of an attribute is abstracted to a more general value. This effectively reduces the information, but does not remove it entirely. For example a datum with an address field could generalize the exact location to a broader one. Instead of Berlin it would read Germany as shown in Figure 7. This could then be even further generalized to Europe and so forth. Typically, entire generalization hierarchies are provided as parameter to facilitate this. It is crucial that these hierarchies are exhaustive of

all possible arising values if no default is provided as genarlization is ambiguous. These complete generalization hierarchies are required as parameters in addition to their respective attribute key.

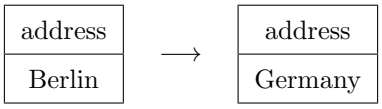


Figure 7: Example of the generalization of an attribute.

A special case of Generalization is **Bucketizing**. It functions similarly, but exclusively for numerical values. Ranges replace specific values in the tuple. Figure 8 shows an example where the age 27 is bucketized to the range [20 - 30]. Only a moderate amount of information is lost with this masking function. To deploy it requires the bucket sizes as well as the keys to the numerical attribute.

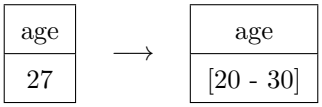


Figure 8: Example of bucketizing of an attribute.

Finally, there are the **Noise Methods**. Again, these only apply to numerical data. The idea is to modify the original values by adding noise. Typically, this noise is chosen randomly from a distribution. Utilizing the normal distribution with mean 0 it would ensure that the data over a period of time would retain its average and distribution. The standard deviation will then define how much each individual data point can diverge from the original value. The result will invalidate individual tuples but preserve the overall spread of the data in the long run. As an example Figure 9 shows noise added to two attributes of a tuple. Note that with no loss to the generality one is decreased, while the other increased.

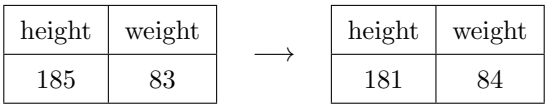


Figure 9: Example of adding noise.

3.3 System Requirements

3.3.1 Data Stream Integration

3.3.2 Administration

3.3.3 Performance

3.3.4 Adaptability

3.3.5 Scalability

3.3.6 Reliability

4 Implementation

4.1 Apache Kafka

4.2 Apache ZooKeeper

4.3 Anonymized Kafka

Class Diagram

4.3.1 Stream Manager

4.3.2 Configuration parsing

4.3.3 Validation

4.3.4 Stream Config Builder

4.3.5 Kafka Streams

4.3.6 Anonymizers

4.4 Test Suite

4.4.1 Data Generator

4.4.2 Kafka Connector

4.4.3 Consumers

4.5 Docker

Putting it all together Docker compose Network Volumes Dependencies Individual Dockerfiles
... should include the following:

- research methodology (e.g., prototype and experiments, case study, literature survey, theoretical analysis),
- derivations and descriptions of algorithms, hardware, software, and/or systems developed.

Algorithm 1: Splitting a Session[7].

```
Parameters:
        e : Tuple to be inserted.
        te (e) : Event-time of e.
S ← slice that covers te (e);
if S starts at te (e) then
    //Slice before S must be fixed.
    change the type of the slice before S to combined;
    add e to S;
else
    // S does not start at te (e).
    change tend (S) to te (e) (excluding te (e) from S);
    change type of S to flexible;
    add slice in [te (e), former tend (S)] with former type of S.
    add e to the new slice.
end
```

5 Testing and Evaluation

TODO

5.1 Experimental Setup

... should include the following:

- define experimental data and workload(s),
- discussion about the selection and interpretation of the evaluation metrics,
- discussion about the computing environment, including hardware, software, tools.

5.X Design and an Interpretation of the Results (For each Experiment Class X)

... should include the following:

- which experiments will be conducted and why?
- for each experiment, what are objectives, baselines, and expected results?
- description and an interpretation of the experimental results,
- explanation for any anomalies or any unexpected behavior.

6 Conclusion

6.1 Future Work

... should include the following:

- problem restated and a brief summary of the methodology,
- student contributions (e.g., survey, open-source software, journal publication),
- a brief summary of the findings and results,
- limitations and generalizability of the findings and results.
- lessons learned,
- recommendations for future research.

Bibliography

[1] Cao, J., Carminati, B., Ferrari, E., Tan, K.L.: Castle: A delay-constrained scheme for k-anonymizing data streams. pp. 1376–1378. IEEE (4 2008)

[2] Federation, I.D.: Diabetes facts & figures (2023), https://idf.org/about-diabetes/diabetes-facts-figures/, accessed: 2023-10-01

[3] Gévay, G.E., Rabl, T., Breß, S., Madai-Tahy, L., Quiané-Ruiz, J.A., Markl, V.: Efficient control flow in dataflow systems: When ease-of-use meets high performance (2021), https://www.researchgate.net/publication/349768477_Efficient_Control_Flow_in_Dataflow_Systems_When_Ease-of-Use_Meets_High_Performance, to be published

[4] Kafka, A.: Organizations powered by apache kafka (2023), https://kafka.apache.org/powered-by, accessed: 2023-10-01

[5] Mahdi Esmailoghli, Jorge-Arnulfo Quiané-Ruiz, Z.A.: Cocoa: Correlation coefficient-aware data augmentation (2021)

[6] Organization, W.H.: Diabetes fact sheet (2023), ttps://www.who.int/news-room/fact-sheets/detail/diabetes, accessed: 2023-10-01

[7] Traub, J., Grulich, P.M., Cuéllar, A.R., Bress, S., Katsifodimos, A., Rabl, T., Markl, V.: Scotty: General and efficient open-source window aggregation for stream processing systems (2020), https://www.redaktion.tu-berlin.de/fileadmin/fg131/Publikation/Papers/Traub_TODS-21-Scotty_preprint.pdf

[8] Wallwork, A.: English for writing research papers, chap. Abstracts, pp. 177–245. Springer International Publishing Switzerland (2011)

[9] Wallwork, A.: English for writing research papers, p. 306. Springer International Publishing Switzerland (2011)

Appendix A. Further Details on the Solution Approach

Appendix B. Extended Version of the Experimental Results