Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.052310

ARTICLE

Enhancing AI System Privacy: An Automatic Tool for Achieving GDPR Compliance in NoSQL Databases

Yifei Zhao, Zhaohui Li and Siyi Lv*

College of Cyber Science, Nankai University, Tianjin, 300350, China

*Corresponding Author: Siyi Lv. Email: lvsiyi@nankai.edu.cn

Received: 29 March 2024 Accepted: 21 May 2024 Published: 18 July 2024

ABSTRACT

The EU's Artificial Intelligence Act (AI Act) imposes requirements for the privacy compliance of AI systems. AI systems must comply with privacy laws such as the GDPR when providing services. These laws provide users with the right to issue a Data Subject Access Request (DSAR). Responding to such requests requires database administrators to identify information related to an individual accurately. However, manual compliance poses significant challenges and is error-prone. Database administrators need to write queries through time-consuming labor. The demand for large amounts of data by AI systems has driven the development of NoSQL databases. Due to the flexible schema of NoSQL databases, identifying personal information becomes even more challenging. This paper develops an automated tool to identify personal information that can help organizations respond to DSAR. Our tool employs a combination of various technologies, including schema extraction of NoSQL databases and relationship identification from query logs. We describe the algorithm used by our tool, detailing how it discovers and extracts implicit relationships from NoSQL databases and generates relationship graphs to help developers accurately identify personal data. We evaluate our tool on three datasets, covering different database designs, achieving an F1 score of 0.77 to 1. Experimental results demonstrate that our tool successfully identifies information relevant to the data subject. Our tool reduces manual effort and simplifies GDPR compliance, showing practical application value in enhancing the privacy performance of NOSQL databases and AI systems.

KEYWORDS

GDPR compliance; NoSQL databases; AI system; privacy

1 Introduction

The world's first major act to regulate Artificial Intelligence, the AI Act [1], was passed by the European Union on 13 March 2024. Article 2 of the act requires that AI systems' processing of personal data must comply with the General Data Protection Regulation (GDPR) [2]. For providers of artificial intelligence system services, achieving GDPR compliance is an important task, as non-compliance can result in severe penalties, including hefty fines and extensive mandatory audits. GDPR affords users the right to issue Data Subject Access Requests (DSARs). An individual can submit a DSAR to an organization to modify or delete their personal information or obtain a copy of their data.

Upon receiving a DSAR, the organization must provide the relevant information or take the requested action.

To comply with DSAR for achieving GDPR compliance, organizations need to track metadata based on user requests to provide complete data related to personal information. Database administrators (DBA) need to identify which tables in the database contain information related to a specific user and which ones do not. For instance, supposing a simple transactions system, the *user* table in the database stores user information. The *accounts* table stores the user's account information, while another table named *transactions* holds transaction records for the account. According to GDPR, users have the right to request the data controller delete their personal information, including transaction records and accounts. In Relational Database Management System (RDBMS), when a DSAR is received, The DBA must query the tables connected to *user* via either direct foreign keys, such as *accounts*, or via indirect ones, such as *transactions*. When the database lacks foreign key constraints, techniques like foreign key inference can complement these relationships, assisting DBAs in information extraction and reducing the likelihood of errors in purely manual operations [3]. The existing strategies for compliance can be summarized into three types. The most direct path is to redesign new data management systems to meet the requirements. The second path is to retrofit existing systems with the additional functionality necessary for GDPR compliance, and the third choice is to use external tools to retroactive compliance.

Since AI systems often need to handle large amounts of unstructured data, the distributed architecture and flexible data model capabilities of NoSQL databases may be more suitable for storing and managing this data. Unlike RDBMS, NoSQL databases are non-relational, schema-less, and do not have foreign key constraints. In NoSQL databases, the weak association between data is emphasized to achieve scalability. Although commonly used NoSQL databases provide interfaces for table joins and queries, determining which data to return depends on the specific application's use case. In the absence of foreign key associations and the inability to use foreign key inference, developers must manually identify information and write custom queries based on semantics when receiving a DSAR, leading to a higher risk of non-compliance. Thus, there is a need to explore how to automate this process. This process faces three challenges.

Firstly, the connections between data may be indirect and implicit. For example, in the transaction system, the association between the *user* table and the *transactions* table requires the *accounts* table to serve as an intermediary. In the NoSQL database, there is no explicit constraint reflecting this relationship. Even with direct connections, the DBA can only write queries based on semantic understanding rather than explicit foreign key constraints. In a document-oriented database, a document's internal collection may have multiple embedded structures. While this flexible data model allows developers to store data with different structures (unstructured data), it may make the relationships between the data even less apparent, making precise data identification and extraction challenging.

Secondly, the response to DSAR requires the DBA to retrieve all data associated with the user from the database. If a user requests the deletion of their personal data, but the DBA fails to delete all personal information, it can lead to non-compliance. However, applications often have data redundancy for performance reasons. To reduce query time, the same data may have multiple copies. In NoSQL databases, the use of a large number of embedded structures makes it more difficult to extract complete information.

Thirdly, Responding to DSAR requires avoiding extracting too much data. Extracting too much data can also lead to non-compliance. Some data, although related to individuals, may fall outside the scope, such as data that is not in compliance with legal requirements or involves the privacy of other data subjects.

This paper addresses these challenges. Our work is based on MongoDB [4], a typical document-oriented database, aiming to generate a relationship graph to represent data relationships in NoSQL databases visually. Then, generate a series of queries according to this relationship graph. Our work is primarily based on three ideas: Firstly, we use the form of a relationship graph to represent the implicit relationships between data. We believe that relationship graphs are helpful in identifying data relationships in NoSQL databases, thereby assisting developers and DBAs in responding to DSAR. Secondly, we present a reverse engineering strategy based on function dependency detection and schema inference. This approach is heuristic. It builds relationships between various data entities by detecting explicit references, implicit references, and data aggregation present in each document. We can traverse databases and establish relationships between various data entities. Database logs can also help infer relationships. This relationship establishment is at the application layer. If the application associates two data entities, we can infer that there is a relationship between these two pieces of data. Thirdly, we provide customizations based on the automatically generated relationship graph to assist DBAs in handling data to avoid extracting too much data.

Based on these ideas, we developed a tool for responding to DSAR. Of the three compliance strategies mentioned, our tool falls into the third path. We believe this approach is the most practical option, which would allow an organization to fulfill regulatory requirements while retaining its current data management system. Previous research work in this approach was limited to RDBMS. Our tool targets more general data schemas, thereby better enhancing the privacy compliance of AI systems. We evaluated our tool using three databases: 1) *sample_flix*, a MongoDB-provided sample dataset; 2) *Pubs*, a sample dataset provided by Studio 3T; and 3) *HRIS*, a human resources management system database. We manually identified all collections involved in responding to DSAR as ground truth. After evaluation, we found our tool to be effective on these databases.

In summary, this paper makes the following contributions:

1. We developed a tool to help DBA extract data satisfying DSAR from NoSQL databases. This tool takes JSON files (which store data from the database) and database query logs as input. It automatically generates a relationship graph and a set of queries.
2. We describe the algorithm used by our tool, including how to discover and extract relationships from a NoSQL database and how to generate a relationship graph.
3. We evaluate this tool on three databases, demonstrating the practicality of this approach. *sample_mflix* is used for sanity checks, *Pubs* and *HRIS* are used to evaluate how our tool performs in databases with different designs.

The rest of this paper is organized as follows. Section 2 introduces the relevant background of our work. Section 3 then details the remaining work in the same field related to our work. Section 4 introduces the difficulties faced in our work and our solutions. Section 5 further elaborates on the specific technical details. Experiments and evaluations are conducted in Section 6. Section 7 summarizes our work and proposes future research directions.

2 Background

Artificial Intelligence Act: The Artificial Intelligence Act (AI Act) [1] is a European Union regulation on artificial intelligence in the European Union. Proposed by the European Commission on 21 April 2021 and passed on 13 March 2024, it aims to ensure that AI technologies are developed and used in a way that is ethical, transparent, and respects fundamental rights. The Act includes provisions for the governance of AI systems, risk assessment, data governance, and accountability mechanisms. It also establishes requirements for high-risk AI systems, such as those used in critical infrastructure, healthcare, or law enforcement, to undergo conformity assessments before they can be placed on the market or put into service. The AI Act is part of the EU's broader strategy to promote trustworthy AI and protect individuals from potential harms associated with AI technologies.

Privacy Laws. General Data Protection Regulation (GDPR) is a comprehensive data protection law that came into effect on 25 May 2018, within the European Union (EU) [2]. This regulation was designed to harmonize data privacy laws across Europe, protect the data rights of EU citizens, and reshape the way organizations across the continent approach data privacy. The GDPR not only applies to companies based within the EU but also to those outside the EU if they process the personal data of EU residents. One of the primary objectives of the GDPR is to give individuals more control over their personal data. To this end, it introduces several rights for data subjects—the individuals whose personal data is being processed. These rights include the right to access, rectify, erase, restrict, port, object to the processing of their data, and more.

GDPR has become a model for other privacy laws. With the growing concern of citizens about personal privacy information, new privacy regulations continue to be introduced [5–8]. These privacy regulations also grant users rights over their personal data and impose requirements for data processing.

Data Subject Access Request. A crucial aspect of GDPR is the Data Subject Access Request (DSAR). DSAR is a mechanism that empowers individuals to inquire about whether, why, and how a company is processing their personal data. Upon such a request, organizations are obliged to provide: Confirmation that the individual's data is being processed. Access to the actual personal data being held. Any available information about the logic involved in automated decisions, if applicable. Organizations are required to respond to DSAR without undue delay and, in any case, within 1 month of receiving the request. This period can be extended under certain conditions, such as the complexity of the request. In June 2023, Sweden's data protection authority fined internet music and podcast streaming giant Spotify 58 million Swedish Kroner (about $ 5.4 million) for deficiencies in how it responds to customer DSAR [9]. How to better respond to DSAR is a very practical need for organizations.

NoSQL Database. NoSQL refers to "not only SQL" or "non-relational" and broadly describes non-relational databases. These databases are schema-less and utilize non-relational data models. NoSQL databases can often be easily scaled by adding more servers to the existing infrastructure. They are typically better suited for handling large volumes of unstructured or semi-structured data. Thus, their market share has been increasing year by year. There are primarily four types of NoSQL databases:

1. Key-Value Databases: This type uses a key-value pair format to store data, where the key is a unique identifier, and the value can be structured, semi-structured, or unstructured data.
2. Document Databases: Each entry is a document that can contain multiple key-value pairs, arrays, or embedded documents. This makes document databases suitable for more complex and flexible data models.
3. Column-Oriented Databases: The data structure is organized and stored in the form of column families, mainly used for distributed file systems.
4. Graph Databases: Graph databases are specifically designed for storing graph-structured data.

3 Related Work

To achieve GDPR compliance, researchers have conducted extensive studies, which can be categorized into three types of measures. We summarize these research findings and analyze their characteristics.

First, one approach is to proactively design new data management systems that achieve privacy compliance by construction. To our knowledge, the research by Schwarzkopf et al. was the first to explore this approach [10]. This approach incurs minimal performance loss, aiding future deployments, but do not assist old databases in meeting GDPR compliance. Updating an entire database system is a time-consuming and laborious task for organizations. A compliant-by-construction system should prioritize data ownership and provide DBAs and developers with an easy-to-understand API to identify users' personal data to comply with privacy regulations. SchengenDB [11] and K9db [12] are new explorations in this direction.

The second approach involves retrofitting existing systems, such as adding metadata structures and secondary indices to existing database systems. Shastri et al. studied this strategy and found that despite requiring a modest number of changes to storage systems, GDPR compliance results in significant performance overhead [13]. Strict metadata tracking incurs a 2–5x performance overhead [14]. Therefore, István et al. suggested accelerating these operations in hardware [15].

The third approach is to use external tools to retroactive compliance, allowing organizations to meet compliance requirements while retaining their original database systems, with a moderate additional performance overhead. This is also the approach used in our paper. However, this method also has its drawbacks, as it still requires a certain proportion of manual operations to achieve compliance. Some application-specific compliant plugins fall into this category. However, these plugins lack universality [16–19]. Prior to our work, odlaw [20] and GDPRrizer [3] used this approach. odlaw assumes that database schemas contain explicit foreign keys, which many applications lack. Additionally, both studies are limited to RDBMS and cannot be applied to NoSQL databases. Table 1 shows the three strategies along with their respective strengths and weaknesses.

Table 1: The advantages and disadvantages of the three approaches for GDPR compliance

Strategy	Advantages	Disadvantages
Redesign new system	• Minimal performance loss • Can meet most requirements	• Not suitable for legacy databases • Costly transition for organizations • May not remain robust to future regulatory changes
Retrofit existing systems	• Clear execution logic • Suitable for legacy databases	• Significant performance overhead • Still requires substantial changes
Use external tools	• Suitable for legacy databases • Can be flexibly adapted to multiple applications	• Requires manual operations • May harm performance

In addition, there are some studies relevant to our work. The development of artificial intelligence systems has raised concerns about privacy and data security. Some researchers have described the issue of personal information leakage in large language models (LLM) [21,22]. Sebastian emphasized the importance of continued research, regulation, and application of Privacy-Enhancing Technologies

(PETs) in artificial intelligence models [23]. Our algorithm is based on discovering implicit data dependencies in NoSQL databases. There are many studies related to this topic. Gómez et al. analyzed the importance of data schemas for NoSQL systems [24]. Mior et al. proposed a semi-automatic technique to extract a normalized model from an existing non-normalized database [25]. Further research presented a functional and inclusion dependency detection method in JSON files [26].

4 Problem Statement

4.1 Difficulties in Complying with DSAR

In real application scenarios, compliance with DSAR is challenging due to the complexity of data ownership and semantics in databases. Precisely identifying data relevant to a data subject is a manual and difficult task. We summarized them into three challenges.

Challenge 1: Determine which data is relevant to the data subject.

According to Article 4 of the GDPR [2], 'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person.

After receiving a DSAR, the first task of DBAs is to determine which information in the database is relevant to the data subject based on the definition of personal data. In a relational database management system (RDBMS), such data relationships are often explicitly represented by foreign keys. Data extraction requires first identifying how data is cross-referenced across tables and then returning the personal information associated with the data subject in the user table.

However, some data in certain tables may be indirectly linked to the data subject through one or more intermediary tables rather than through direct foreign key relationships. Additionally, not all databases enforce foreign key constraints rigorously. The database schema of real-world applications evolves over time and can become quite messy. The current solution for such issues is to use foreign key inference to supplement the database with the foreign key constraints it should have had, thereby obtaining a complete data retrieval.

In NoSQL databases, this problem becomes even more complex. NoSQL databases store data without a predefined schema. This schema-less approach provides developers with more flexibility in data storage. However, it also means that NoSQL databases lack explicit relationships like foreign key constraints. Relationships between data are implicit, requiring DBAs to rely on semantic understanding to determine which data is related to the data subject.

Challenge 2: Ensure all personal data is extracted.

Organizations receiving DSAR should ensure that all information related to the data subject in their databases is correctly identified. Failure to delete all information when a data subject requests the erasure of their personal data can lead to potential privacy leaks and penalties. The presence of redundant data in a database can compromise its integrity. In RDBMS, well-formed database schemas in 3NF can reduce data redundancy. However, in practical applications, the normalization process when designing a database may not be perfect. Some data may be replicated across multiple tables to reduce the number of join operations for improved query performance.

The characteristics of NoSQL databases make them more prone to data redundancy compared to relational databases with strict constraints. NoSQL databases always handle large-scale data and high-concurrency access. In a document, two objects may have an aggregation relationship, but the child object may also have a duplicate copy of the same content stored in another document. This design choice is made for better performance. When there is a relationship between two documents, developers often choose to store data using embedded documents to avoid cross-document access. When responding to DSAR, it is necessary to find all related data in the database, which requires traversing the entire database and extracting all information associated with a specific user.

Challenge 3: Avoid extracting too much data.

Some data, though related to individuals, should not be disclosed to the data subjects in response to requests. In real-world scenarios, databases often contain information about multiple data subjects. Therefore, when responding to DSAR, it is crucial to avoid disclosing personal information pertaining to other individuals. For instance, Facebook's privacy policy specifies that while the platform deletes comments made by data subjects, it does not delete private messages sent to friends unless those friends also delete them. Additionally, there are types of data associated with data subjects that organizations are required to retain even after receiving deletion requests, owing to legal and public interest considerations. Furthermore, certain data should not be disclosed in response to DSAR due to specific requirements of the application's use case. Even when two databases share the same structure, the data that needs to be disclosed in response to DSAR may vary based on the application's scenarios. GDPR provides organizations with some degree of flexibility in handling DSAR. Understanding how to handle personal data correctly to address practical scenarios is another crucial aspect of compliance with data protection laws.

4.2 Approaches

To address Challenge 1, for indirect data associations and implicit dependencies, a relationship graph can be used to assist analysis. The vertices of the graph consist of data entities and the relationships between data entities form the edges of the graph. The absence of an explicitly declared schema does not imply a lack of schema, as the schema is implicit in the data and the database. In NoSQL databases, the terms aggregation and reference are used to describe relationships between data objects. An aggregation structure refers to the organization of related data as a cohesive unit within a single document, often involving embedded documents or arrays representing hierarchical relationships. This structure is common in NoSQL databases, and we consider that there is a relationship between two objects that form an aggregate structure. Another type of structure is reference. A reference implies that the pair of an entity identifies an object from another entity. NoSQL databases often use distributed cluster architectures for scalability. So, reference structures are often not the optimal choice due to potential remote node access. We consider two objects with a reference relationship to be represented as an edge in the relationship graph. This operation is based on the assumption that if there is a reference relationship between two data entities, they are semantically related. If one of the data entities is a data subject under privacy regulations, we believe it needs to be returned in the response to the DSAR. To infer reference relationships between data entities, we use some data-driven heuristics.

To address Challenge 2, we construct the relationship graph from an application perspective rather than being specific to a particular data entity. The relationship identification algorithm takes the entire database as input, traversing all data to avoid missing any information. The inferred relationship graph often fails to fully represent the data relationships within the entire database, as manual customizations at the graphs level are often necessary. While relationship-discovery algorithms can identify most relationships, even a small amount of missing data can lead to non-compliance. Manually completing the missing relationships in the graph can better comply with DSAR.

For Challenge 3, after manually completing the relationship graph, further output filtering is required. Exclude data that should not be output, such as information related to other data subjects, without altering the relationship graph. We believe that even though we use automated tools to assist in responding to DSAR, complete automation is unlikely. However, a data relationship graph generated in reverse from a NoSQL database, along with queries generated on this basis, can greatly reduce manual effort.

5 Our Scheme

5.1 Overview

Our tool is based on the leading document-oriented database MongoDB [4]. It infers the database schema and data relationships from documents, combined with the results of the database query log, and generates a relationship graph to represent semantic relationships between different objects in the database. We take the userid representing a specific data subject in the application as input to our program and return a relationship graph. Then, generate a set of queries based on it. However, as we mentioned, achieving privacy compliance through external tools still requires some manual operation. Our tool enables DBAs to intervene and customize either the relationship graph itself, or the data output. Fig. 1 shows an overview of our tool. We implemented our tool on MongoDB, but this does not mean that our proposed solution can only be implemented on MongoDB. Our algorithm does not rely on MongoDB-specific features, but only uses MongoDB's query language in the process of generating queries. This means that our tool can be easily ported to other NoSQL databases.

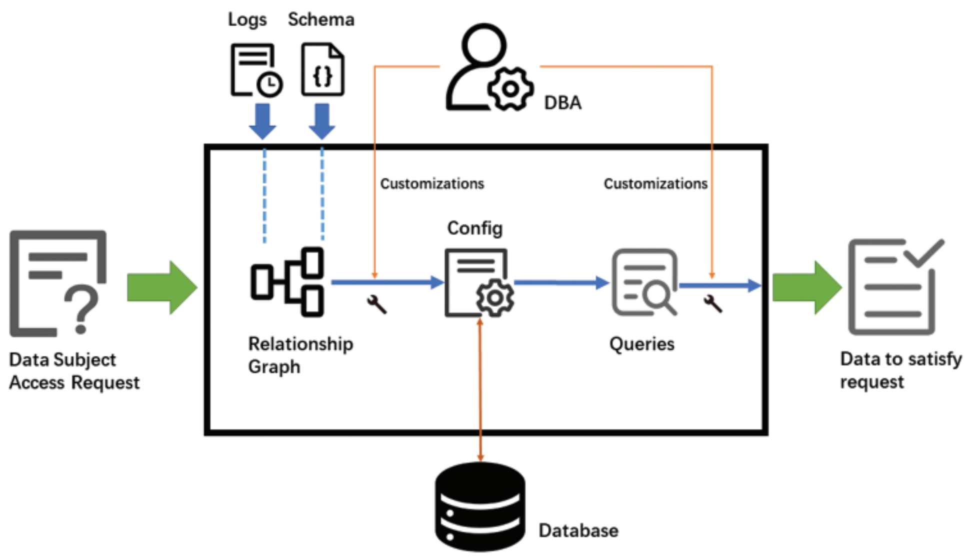


Figure 1: Overview of our tool

5.2 Automated Relationship Detection

In NoSQL databases, a document typically stores information about a data entity. Although the flexibility of document-oriented databases allows for storing multiple types of data within a single document, in practical application, it is believed that different data entities, such as user information and product information, are stored in separate documents. The semantic relationships between different entities in the database are indicated through aggregation and references. Therefore, aggregation and reference are considered as the basis for relationship detection. Our tool takes documents as input and associates data entities by discovering database aggregation and reference.

Another reliable source of relationships is the query log of the database [27]. MongoDB provides aggregation operations for processing multiple documents and returning results. In actual application operations, connecting multiple documents usually implies a relationship between them. Obtaining query logs is straightforward, as commonly used NoSQL databases generally provide a logging system to record and store query logs.

Currently, numerous studies explore the use of reverse engineering strategies to infer implicit schemas in NoSQL databases [28,29]. Such studies typically involve Map-Reduce operations to identify objects with similar structures in documents, using one object as a prototype and getting its raw schema. Due to the scalability of NoSQL databases, this step may result in multiple objects with slightly different structures. As applications are updated, different structures may emerge for the same data entity.

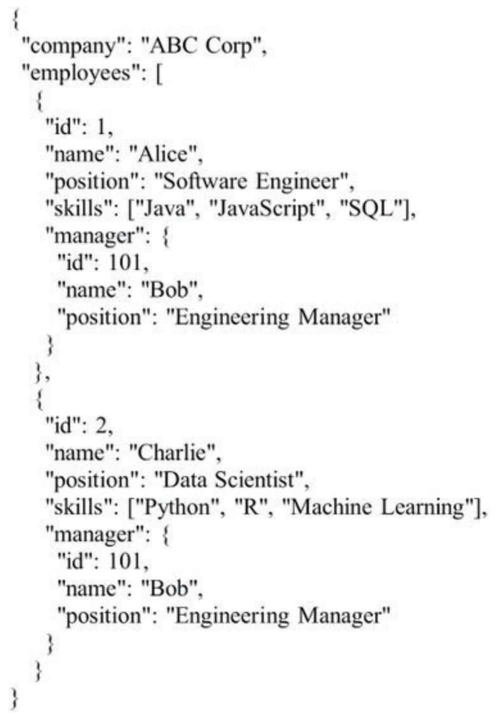
We are not concerned with different data schemas. After the Map-Reduce operation, we integrate schemas with different structures under the same data entity because we assume that one document in the database only stores one data entity. The structural differences arise due to versioning changes. This schema includes all attributes of the data entity under various versions and structures.

5.2.1 Discovering Aggregation Relationships

If the value of a pair is an object or an array of objects, it creates an aggregation relationship. In document-oriented databases, document embedding and aggregation are common. Detecting aggregation between documents is an intuitive operation, whether a single-layer or multi-layer embedding: it only requires checking the type of the value of pairs. When an aggregation relationship is detected, the pair's key is considered a candidate vertex for the relationship graph.

If the detected aggregation relationship has an array as the pair's value, the array usually contains a series of objects with similar or identical structures. We perform another Map-Reduce operation for this sub-document. For example, in a document *transactions*, a pair may store many objects with similar structures representing different transaction records. In this case, the objects generated after Map-Reduce with different structures are considered candidate vertices for the relationship graph. When it is discovered that there is a reference relationship between the sub-documents and other documents, we will add the candidate vertex and the aggregation structure to our relationship graph. Since embedded structures are common in document-oriented databases, many contents in sub-documents are not data entities but can be understood as attributes of the data entity represented by the parent document.

For example, Fig. 2 simulates a *company* document of a human resources management system database. In this document, the value of the *employees* key is an array of multiple objects. We reduce this array to a sub-document, and within this sub-document, there is an embedded object *manager*. Both *manager* and *employees* will have reference relationships with other documents (the documents that store employees' information). Therefore, we consider them as vertices in the graph. And according to the aggregation relationship, draw the edges in the graph like *company*-*employees*-*manage*.



```
{
 "company": "ABC Corp",
 "employees": [
  {
   "id": 1,
   "name": "Alice",
   "position": "Software Engineer",
   "skills": ["Java", "JavaScript", "SQL"],
   "manager": {
    "id": 101,
    "name": "Bob",
    "position": "Engineering Manager"
   }
  },
  {
   "id": 2,
   "name": "Charlie",
   "position": "Data Scientist",
   "skills": ["Python", "R", "Machine Learning"],
   "manager": {
    "id": 101,
    "name": "Bob",
    "position": "Engineering Manager"
   }
  }
}
```

Figure 2: Example for aggregation relationships

5.2.2 Discovering Reference Relationships

MongoDB automatically creates a unique identifier of type *Objectid* for each document. References can be established by referring to *Objectid*. The implementation of references may take different forms depending on the strategy chosen by developers. MongoDB provides a convention called DBrefs [4], using a fixed format to represent references. DBRef documents resemble the following format:

$ref: The name of the collection.

$id: The *_id* field value of the referenced document.

$db: The name of the database. This field is optional and is typically used only when referencing documents in different databases.

If such a structure is detected, it can be determined that the database has declared a reference in an explicit way. Additionally, if a pair name is the name of an existing entity, and the pair values correspond to the values of an *_id* pair of such an entity, we can infer that there is a reference relationship between these two data entities. These two methods address the general reference detection problem. We have extended this approach using a data-driven heuristic method to infer reference relationships between data entities in different documents based on this idea. This approach is inspired by foreign key inference in relational databases [30–32]. Rostin et al. have studied the problem of inferring foreign key constraints in database instances lacking foreign key constraints [33]. This approach can inspire relationship inference in NoSQL databases. We consider that the identifier values of key-value pairs indicating reference relationships should store the unique identifier of another document. In MongoDB, unique identifiers can be created by creating unique indexes. We traverse existing databases' documents, considering all keys with non-repeating values as candidate unique identifiers. Subsequent relationship analysis will be based on these identifiers.

Definition. Inclusion Dependency (IND) Given two relations R_1 and R_2, and attribute sets A in R_1 and B in R_2 , if every value in A in R_1 also appears in B in R_2, we say the values of A are contained in B, represented as $R_1[A] \subseteq R_2[B]$.

If two entities in two documents have a reference relationship, the inclusion dependency of their key-values will be a necessary feature. Detecting inclusion dependency is a deterministic algorithm [34], but the inclusion relationship of this value set may be entirely coincidental. For instance, if the value type of a key-value pair is boolean or its value range is a relatively small integer interval, it will almost certainly create an inclusion dependency. Therefore, based on the detection of inclusion dependency, it is necessary to convert it into an actual reference relationship using some basic features at the time of reference relationship creation, such as the previously mentioned similarity in key names. Rostin et al. proposed ten typical properties of database foreign keys to filter containment dependencies into inferred foreign key constraints. Let $s(A)$ and $s(B)$ denote the set of distinct values of A and B, and let $name(A), name(B)$ denote the attribute names. Based on these ten properties and the characteristics of NoSQL databases, we selected four as our heuristic methods:

1. *TypicalNameSuffix:* Check whether $name(A)$ ends with a substring that is an indication of relationships (e.g., "id").
2. *OutOfRange:* Calculate the percentage of values in B that are outside of $[min(s(A)), max(s(B))]$. The dependent values are always evenly distributed over the referenced values.
3. *KeyName:* Calculate Jaro similarity between $s(A)$ and $s(B)$ to determine if the key names of two keys have sufficient similarity.
4. *Coverage:* Calculate the ratio of values in $s(B)$ that are contained in $s(A)$ to all values in $s(A)$. Keeping pairs that exceed a threshold.

If a pair of candidates with containment dependency passes the filters, we add it as an edge to the relationship graph.

5.2.3 Log Queries

MongoDB offers a powerful query language, including the aggregation pipeline, which allows complex transformations and computations on data. MongoDB provides structured JSON format files as the output for logs. Our tool accepts this file as input and parses logs involving the MongoDB aggregation pipeline. Relationships can be built between different collections based on the content of the pipeline operations.

5.3 Graph Traversal

Generating a set of queries based on the relationship graph is an engineering problem. NoSQL databases lack a universal query language, but various database types often provide specialized statements for query operations. We have written code for MongoDB to traverse the graph and generate queries. The user specifies the *userid* from the user table as the input to the graph. We traverse the entire graph using a breadth-first search method to generate a set of queries. This step is primarily for evaluation. Two issues are involved in the specific traversal process. First, during the traversal process, in addition to the edges obtained by the relationship recognition algorithm, different keys in the same document, which are not connected by edges in the relationship graph, are considered connected. Second, during the traversal process, there may be more than one path connecting two documents. In this case, because we have chosen the BTS method, we will select the shorter path to generate the query. Intuitively, keys closer to the *userid* are considered more relevant to the data subject.

While traversing the graph, our tool extracts the required data. Starting from the userid, for each associated edge, data is extracted using MongoDB's aggregation pipeline operations. An important issue to note is that MongoDB typically contains a lot of duplicate data. In the process of responding to a DSAR, all this data should be extracted. Assuming the user's DSAR is to delete their personal data, extracting too little data would leave copies of it in the database, potentially leading to non-compliance. Because our traversal process will traverse all documents associated with a given id, we can avoid extracting too little data. A notable concern is that traversing the entire database incurs additional performance overhead. However, since this operation is only required once during configuration, we consider this overhead acceptable.

5.4 Customizations

Based on the automatically generated relationship graph, the DBA can customize either the relationship graph directly or the data output post-traversal. Our tool provides two strategies for the relationship graph: adding or pruning edges. The relationship graph generated by the relationship recognition algorithm may miss some edges or incorrectly add some edges. For example, if two keys in the database have similar value ranges, the relationship-discovery algorithm may incorrectly associate these two keys. Additionally, two semantically related keys may not be linked for various reasons, such as non-standard naming conventions. Customization at the relationship graph level can allow the relationship graph to represent semantic relationships in reality better.

A well-constructed relationship graph can generate a set of queries after traversal, but this set of queries may require further output filtering, such as removing information from other data subjects. Our tool also provides customization at the output level. This customization is specific to the database; once completed, DSAR for this database can reduce manual work, thus better meeting GDPR compliance requirements.

6 Evaluation

We prototyped our tools. It takes multiple JSON files exported from the document database and the query log from MongoDB as input. It returns a relationship graph and a set of queries that conform to MongoDB's aggregation pipeline operations. We test our tool on three example datasets. Our evaluation aimed to answer the following questions:

1. Can our tool correctly extract data satisfying DSAR and generate a relationship graph through reverse engineering strategies?
2. To what extent can the automatically generated queries replace manually written queries?
3. After further manual processing, can our tool perfectly satisfy DSAR?

Based on the real use cases of each database, we manually wrote queries using MongoDB's aggregation pipeline to respond to DSAR for each database. We used this set of "ground truth" queries as a benchmark to evaluate the results generated by our tool. For each database, the results generated by the tool were compared with manually obtained results, and precision, recall, and F1 scores were calculated.

$$P = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fp}}, R = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fn}}, F1 = \frac{2 \cdot P \cdot R}{P + R},$$

Our tool operates in two rounds. During the first round, it did not utilize the query logs as input; In the second round, it incorporated them. Because we lacked real-world scenarios query logs for each database, we simulated some aggregation pipeline operations as input to our tool to test its feasibility. However, since we manually added these queries, we will not use them as an indicator for evaluation but rather to verify the rationality. So, we use the first round without query logs input as our evaluation standard.

6.1 Datasets

We used three sample datasets to test our tool: *sample_mflix*, *Pubs*, and *HRIS*. Among these, *sample_mflix* served as a sanity check, while the other two datasets were used to evaluate the tool's performance. In this section, we will describe the contents of these three datasets, how we established basic facts, and how we collected their queries.

sample_mflix: The *sample_mflix* is a MongoDB's sample dataset [35]. This database contains data on movies and movie theaters. It has six documents: *comments*, *embedded_movies*, *movies*, *sessions*, *theaters*, and *users*. These documents contain a significant amount of embedded structures. The comments document references *embedded_movies*, *movies*, and *users* documents, with one pair referencing the automatically generated *objectid* by MongoDB for each document and two pairs referencing implicit unique identifiers in the document. This dataset has 95 MB of data, including 185 users. We populated this data into MongoDB and simulated operations, collecting 70 queries. For ground truth, we manually wrote four queries involving four documents.

Pubs: *Pubs* is a sample dataset from Studio3T [36]. This database is focused on the publishing industry, with two data subjects: *employee* and *author*. *Pubs* contains 13 documents with complex reference relationships. The dataset includes 48.7 MB of data, including 5000 authors and 300 employees. We populated the database with this data. However, since this database evolved from an SQL database, there is a lack of embedded structure within each document. We tested our tool using the *ids* from the *employee* and *author* documents as inputs. Additionally, based on real-world application scenarios, we attempted to replicate possible actions, resulting in the collection of 700 queries. The basic facts we compiled are based on authors and employees, each containing 10 and 7 queries.

HRIS: *HRIS* is a Human Resources Information System created by us based on real-world use cases and the general pattern of MongoDB databases. It contains 11 documents, which describe employees, departments, locations, salaries, and so on. This database has both complex embedded relationships and intricate reference relationships. The data we generated contains 305 employees. We tested using the *IDs* from the employee document as input. The ground truth we compiled includes nine queries.

6.2 Relationship Graph

For these three databases, our tool successfully generated the corresponding relationship graphs based on the input. Table 2 measures the correctness of the tool return without customizations. In the *sample_mflix* database, whether through a relationship-discovery algorithm or log queries, complete three edges can be obtained, and four queries can be generated, which is consistent with the ground truth we wrote in response to DSAR based on semantics. Fig. 3 shows the relationship graph automatically generated by our tool for the *sample_mflix* database.

Table 2: Automatically generated queries with high-level results without customization. Pubs1 represents the data subject as the employee, and Pubs2 represents the authors

	With querylog			Without querylog		
	P	R	F1	P	R	F1
Sample_mflix	1	1	1	1	1	1
Pubs1	0.71	1	0.83	0.71	1	0.83
Pubs2	0.85	1	0.92	0.85	1	0.92
HRIS	0.81	0.94	0.87	0.91	0.67	0.77

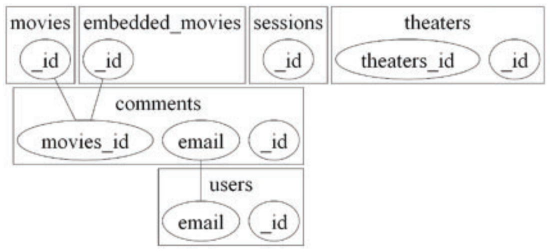


Figure 3: Relationship graph of sample_mflix. sample_mflix has three disconnected components. Each box represents one document and its identifiers

In *Pubs* database, both relationship-discovery algorithm and log queries will lose some edges that should exist. Because *Pubs* has a good naming convention, the relationship-discovery algorithm can better identify the relationships between documents. However, due to the complex reference relationships in *Pubs*, an incorrect connection is established between *stor_id* in the *stores* document and *title_id* in the *titles* document, because the values of these two key-value pairs are almost the same. The performance of our tool will be greatly affected by the database design. Fig. 4 shows the relationship graph for the *Pubs* database. In *HRIS*, the performance of the relationship-discovery algorithm is worse. In the *employee* document, a key named *manager* is used to indicate an employee's manager. For a sub-document in *employee*, this field will establish a connection with the other sub-document in the *employee* document that stores data about other data subjects. This relationship is similar to a recursive foreign key in RDBMS. Our tool did not identify this relationship. The lack of this relationship does not affect the final result because this edge will be removed during manual customizations due to its impact on other data subjects. But we think that this situation still demonstrates the limitations of the relationship-discovery algorithm. Furthermore, high-precision relationship discovery also depends on the size of the data. When the number of sub-documents in the input JSON document is small, relationship recognition will perform poorly.

6.3 Impact of Customizatio

As mentioned, due to the complex scenarios faced by databases in real-world situations, even accurately extracted relationship graphs require manual customizations to meet DSAR. These customizations include adding and pruning edges at the relationship graph level and output filtering at the output level. The *Pubs* document contains two types of data subjects. According to GDPR, we should not disclose the privacy of other data subjects when responding to DSAR from one data subject. Therefore, this leads to the need to filter some data manually. Although our tool identifies almost all reference relationships in *Pubs*, there are still some false positives. In the *HRIS* database, in addition to edge pruning, there is a need to add some edges that have not been correctly identified. In general, edge pruning can improve precision, while edge additions can improve recall.

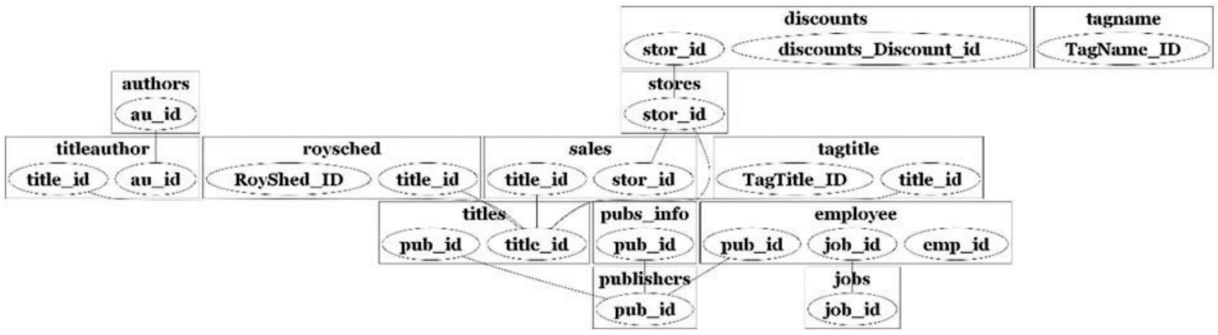
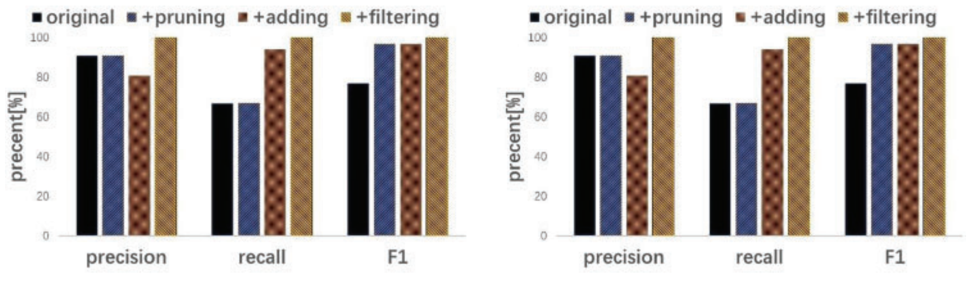


Figure 4: Relationship graph of Pubs

On the basis of a complete relationship graph, output filtering can further improve precision. Because our relationship graph is built from an application perspective rather than being specific to a particular data subject, for data deletion, we use output filtering instead of edge pruning. This includes data that is irrelevant to the user, information related to other data subjects, and information that should not be returned based on actual usage scenarios or legal requirements.

sample_mflix, due to its simple database structure, achieves 100% recall and precision even without customization, as shown in Fig. 5a. *Pubs* achieves 100% recall without customization. After edge pruning, its precision increases. However, due to *Pubs* having two data subjects, even a complete relationship graph results in a loss of precision. After filtering, precision increases to 100%, as shown in Fig. 5b. *HRIS* requires the most complex customization process. Firstly, based on the automatically generated relationship graph, the DBA needs to add and prune some edges to make the relationship graph consistent with the semantic relationship, including the recursive references in the employee document. Customization at the relationship graph level complements the missing edges. However, due to the redundant return of content in the sub-document where the *manager* is located, precision is affected. After filtering the output, precision and recall both increase to 100%. Fig. 6 reflects the impact of customization on output.



(a) Pubs1: employee (b) Pubs2: author

Figure 5: The impact of customization in Pubs. Edge pruning improves precision the most. Only through output filtering can we get 100% accuracy

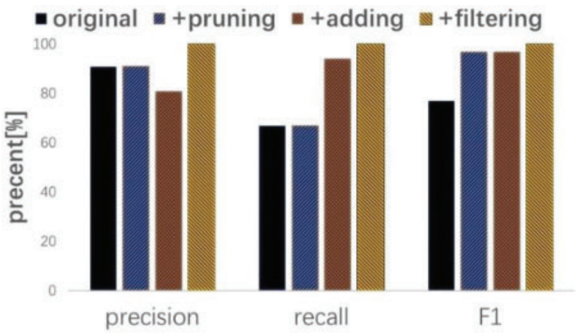


Figure 6: The impact of customization in HRIS. After adding edges, the precision dropped. After passing the filter, the precision reached 100%

Our results indicate that our tool can assist organizations in responding to DSAR in NoSQL databases more easily. The automatically generated relationship graph helps identify relationships between different documents in the database. Through evaluation, we found that the generation of the relationship graph is greatly affected by the normalization of database modeling and the amount of data. For different scenarios and architectures of databases, the results may vary significantly.

7 Conclusion

The AI Act imposes requirements for the privacy compliance of artificial intelligence systems. Organizations must take the response to DSAR seriously. For the increasingly popular NoSQL databases, this work presents a challenge. We propose an algorithm and develop a tool to help database administrators meet these requirements. Since privacy relations are semantically based, the existence of a fully automated tool is improbable. However, through these tools, we aim to reduce the possibility of errors occurring when this process is executed manually. Our future work will further explore automated solutions to assist with compliance. Additionally, our work lacks evaluation in large-scale heterogeneous data. Exploration and validation in this area are another direction of our work.

Acknowledgement: Not applicable.

Funding Statement: This research was supported by the National Natural Science Foundation of China (No. 62302242) and the China Postdoctoral Science Foundation (No. 2023M731802).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Yifei Zhao; data collection: Yifei Zhao; analysis and interpretation of results: Yifei Zhao, Zhaohui Li; draft manuscript preparation: Yifei Zhao, Siyi Lv. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The dataset used in this work is publicly available.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] *Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonized Rules on Artificial Intelligence (Artificial Intelligence Act), COM/2021/206 Final*. Brussels, Belgium: European Commission, Apr. 21, 2021. Accessed: May 18, 2024. [Online]. Available: https://artificialintelligenceact.eu/the-act/

[2] *General Data Protection Regulation (GDPR), Regulation (EU) 2016/679*. Brussels, Belgium: European Parliament and Council of the European Union, Apr. 27, 2016. Accessed: May 18, 2024. [Online]. Available: https://gdpr-info.eu/

[3] A. Agarwal, M. George, A. Jeyaraj, and M. Schwarzkopf, "Retrofitting GDPR compliance onto legacy databases," *Proc. VLDB Endow.*, vol. 15, no. 4, pp. 958–970, Dec. 2021. doi: 10.14778/3503585.3503603.

[4] "MongoDB documents," Accessed: May 18, 2024. [Online]. Available: https://www.mongodb.com/docs/manual/

[5] *The Digital Personal Data Protection Act*. New Delhi, India: Government of India, 2023. Accessed: May 18, 2024. [Online]. Available: https://prsindia.org/files/bills_acts/bills_parliament/2023/Digital_Personal_Data_Protection_Act,_2023.pdf

[6] *Thailand Personal Data Protection Act 2019, B.E. 2562*. Bangkok, Thailand: Thailand Netizen, May 27, 2019. Accessed: May 18, 2024. [Online]. Available: https://thainetizen.org/wp-content/uploads/2019/11/thailand-personal-data-protection-act-2019-en.pdf

[7] *Brazilian General Data Protection Law (Lei Geral de Proteção de Dados Pessoais), Law No. 13,709/2018*. Brasília, Brazil: National Congress of Brazil, Aug. 14, 2018. Accessed: May 18, 2024. [Online]. Available: https://iapp.org/media/pdf/resource_center/Brazilian_General_Data_Protection_Law.pdf

[8] *California Consumer Privacy Act of 2018 (Assembly Bill No. 375), AB-375*. Sacramento, California, USA: California Legislative Information, Jun. 28, 2018. Accessed: May 18, 2024. [Online]. Available: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375

[9] "Administrative fee against spotify," Accessed: May 18, 2024. [Online]. Available: https://www.imy.se/en/news/administrative-fee-against-spotify/

[10] M. Schwarzkopf, E. Kohler, M. F. Kaashoek, and R. Morris, "Position: GDPR compliance by construction," in *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*, Los Angeles, CA, USA: Springer, Aug. 30, 2019, vol. 5, pp. 39–53.

[11] T. Kraska, M. Stonebraker, M. Brodie, S. Servan-Schreiber, and D. Weitzner, "Schengen DB: A data protection database proposal," in *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*, Los Angeles, CA, USA: Springer, Aug. 30, 2019, vol. 5, pp. 24–38.

[12] K. D. Albab *et al.*, "K9db: Privacy-compliant storage for web applications by construction," presented at the 17th USENIX Symp. Oper. Syst. Des. Implement. (OSDI 23), Boston, MA, USA, 2023, pp. 99–116.

[13] A. Shah, V. Banakar, S. Shastri, M. Wasserman, and V. Chidambaram, "Analyzing the impact of GDPR on storage systems," presented at the 11th USENIX Workshop Hot Topics Storage and File Syst. (HotStorage 19), Renton, WA, USA, 2019, pp. 4.

[14] S. Shastri, V. Banakar, M. Wasserman, A. Kumar, and V. Chidambaram, "Understanding and benchmarking the impact of GDPR on database systems," *Proc. VLDB Endow.*, vol. 13, no. 7, pp. 1064–1077, Mar. 2020. doi: 10.14778/3384345.3384354.

[15] Z. István, S. Ponnapalli, and V. Chidambaram, "Software-defined data protection: Low overhead policy compliance at the storage layer is within reach!," *Proc. VLDB Endow.*, vol. 14, no. 7, pp. 1167–1174, Mar. 2021. doi: 10.14778/3450980.3450986.

[16] M. Degeling, C. Utz, C. Lentzsch, H. Hosseini, F. Schaub and T. Holz, "We value your privacy ... now take some cookies: Measuring the GDPR's impact on web privacy," presented at the 2019 Network Distrib. Syst. Secur. Symp. (NDSS), San Diego, CA, USA, Feb. 2019, pp. 24–27.

[17] "GDPR compliance & cookie consent," Accessed: May 18, 2024. [Online]. Available: https://wordpress.org/plugins/gdpr-compliance-cookie-consent/

[18] "The GDPR framework by data443," Accessed: May 18, 2024. [Online]. Available: https://wordpress.org/plugins/gdpr-framework/

[19] "WP GDPR compliance," Accessed: May 18, 2024. [Online]. Available: https://wordpress.org/plugins/wp-gdpr-compliance/

[20] C. Luckett, A. Crotty, A. Galakatos, and U. Cetintemel, "Odlaw: A tool for retroactive GDPR compliance," present at the 2021 IEEE 37th Int. Conf. Data Eng. (ICDE), Chania, Greece, 2021, pp. 2709–2712.

[21] W. Hariri, "Unlocking the potential of ChatGPT: A comprehensive exploration of its applications, advantages, limitations, and future directions in natural language processing," arXiv preprint arXiv:2304.02017, 2023.

[22] G. Sebastian, "Do ChatGPT and other AI Chatbots pose a cybersecurity risk?: An exploratory study," *Int. J. Secur. Priv. Pervasive Comput. (IJSPPC)*, vol. 15, no. 1, pp. 1–11, Mar. 2023. doi: 10.4018/ijsppc.320225.

[23] G. Sebastian, "Privacy and data protection in ChatGPT and other AI chatbots: Strategies for securing user information," *Int. J. Secur. Priv. Pervasive Comput. (IJSPPC)*, vol. 15, no. 1, pp. 1–14, Jul. 2023. doi: 10.4018/ijsppc.325475.

[24] P. Gómez, R. Casallas, and C. Roncancio, "Data schema does matter, even in NoSQL systems!," presented at the 2016 IEEE Tenth Int. Conf. Res. Chall. Inf. Sci. (RCIS), Grenoble, France, 2016, pp. 1–6.

[25] M. J. Mior and K. Salem, "Renormalization of NoSQL database schemas," presented at the Concept. Model.: 37th Int. Conf. ER 2018, Xi'an, China, Springer, Oct. 22–25, 2018, pp. 479–487.

[26] M. J. Mior, "Fast discovery of nested dependencies on JSON data," arXiv preprint arXiv:2111.10398, 2021.

[27] S. Lopes, J. M. Petit, and F. Toumani, "Discovering interesting inclusion dependencies: Application to logical database tuning," *Inf. Syst.*, vol. 27, no. 1, pp. 1–19, Mar. 2002. doi: 10.1016/S0306-4379(01)00027-8.

[28] D. Sevilla Ruiz, S. F. Morales, and J. García Molina, "Inferring versioned schemas from NoSQL databases and its applications," presented at the Concept. Model.: 34th Int. Conf. ER 2015, Stockholm, Sweden, Springer, Oct. 19–22, 2015, pp. 467–480.

[29] M. A. Baazizi, D. Colazzo, G. Ghelli, and C. Sartiani, "Parametric schema inference for massive JSON datasets," *VLDB J.*, vol. 28, no. 4, pp. 497–521, Jan. 2019. doi: 10.1007/s00778-018-0532-7.

[30] J. Bauckmann, U. Leser, F. Naumann, and V. Tietz, "Efficiently detecting inclusion dependencies," presented at the 2007 IEEE 23rd Int. Conf. Data Eng., Istanbul, Turkey, 2007, pp. 1448–1450.

[31] T. Papenbrock *et al.*, "Functional dependency discovery: An experimental evaluation of seven algorithms," *Proc. VLDB Endow.*, vol. 8, no. 10, pp. 1082–1093, Jun. 2015. doi: 10.14778/2794367.2794377.

[32] T. Papenbrock, S. Kruse, J. A. Quiané-Ruiz, and F. Naumann, "Divide & conquer-based inclusion dependency discovery," *Proc. VLDB Endow.*, vol. 8, no. 7, pp. 774–785, Feb. 2015. doi: 10.14778/2752939.2752946.

[33] A. Rostin, O. Albrecht, J. Bauckmann, F. Naumann, and U. Leser, "A machine learning approach to foreign key discovery," presented at the 12th Int. Workshop Web Databases (WebDB), Rhode Island, USA, Providence, Jun. 28, 2009.

[34] M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava, "On multi-column foreign key discovery," *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 805–814, Sep. 2010. doi: 10.14778/1920841.1920944.

[35] "MongoDB atlas," Accessed: May 18, 2024. [Online]. Available: https://www.mongodb.com/docs/atlas/sample-data/

[36] "Studio 3T," Accessed: May 18, 2024. [Online]. Available: https://studio3t.com/