ISSN: 3030-3346

COMPARATIVE ANALYSIS AND IMPLEMENTATION OF MULTI-LEVEL NORMALIZATION TECHNIQUES IN NETWORK DATABASES

Ziyoda Norqulova Nabi qizi[1], Jumanazarov Javlon Shoxnazar o'g'li[1]

[1]Tashkent State University of Economics Tashkent, Uzbekistan
E-mail: ziyodanorqulova@mail.ru, j.jumanazarov@tsue.uz

K E Y W O R D S

multi-level data normalization, network databases, comparative normalization, data types, semantic consistency, ontological mapping

A B S T R A C T

Modern network databases integrate data from heterogeneous sources, which differ significantly in format, scale, encoding, and semantic structure. The challenge of normalizing such data across multiple levels — record, field, and value — remains a critical step in data integration pipelines. This paper presents a comparative analysis and practical implementation of multi-level normalization techniques across twelve distinct data types encountered in network databases: numeric/decimal, integer, text, categorical, temporal, boolean, binary/file, spatial, monetary, array/set, JSON/XML, and null values. For each data type, we systematically compare candidate normalization methods — including min-max scaling, Z-score standardization, one-hot encoding, label encoding, Unicode normalization, ISO 8601 formatting, and log-scaling — identifying the conditions under which each approach is most appropriate. We highlight the trade-offs between methods in terms of distribution preservation, semantic consistency, and computational cost. A Python-based implementation is evaluated on a synthetic dataset of 1,000 records, demonstrating that applying type-aware multi-level normalization reduces numerical attribute dispersion by approximately 70% and achieves 100% structural consistency in text and semi-structured fields. The proposed framework supports downstream tasks including ontological mapping, semantic integration, and 3D data visualization.

Introduction

Currently, large volumes of generated data are stored in various sources –servers, web platforms, social networks, IoT devices, or databases. In each source, data is presented in a specific format, units, coding system, and semantic structure. For this reason, the normalization process is important in data processing.

Normalization is the process of bringing information in different formats into a single structural and semantic form. This process reduces differences between data, making it possible to compare and integrate them. In modern approaches, normalization is not limited to cleaning relational databases, but is also applied to large volumes of many types of (heterogeneous) data. In particular, in network data processing systems, where elements such as nodes, connections, attributes, and weights are present, normalization is an important step.

Scientific literature proposes normalization methods specific to different types of information. Existing surveys tend to treat normalization as a single-level problem, focusing either on numeric scaling or on relational database normal forms, without addressing the multi-level, multi-type nature of real-world network data.

This paper makes the following contributions:

- A systematic taxonomy of twelve data types encountered in network databases, with a description of their normalization challenges.
- A comparative analysis of normalization methods for each data type, including quantitative metrics where applicable.
- A three-level normalization framework (record, field, value) applied consistently across all data types.
- A Python implementation evaluated on a 1,000-record synthetic dataset, with before-and-after performance metrics.

(cc) BY

- Practical recommendations for selecting normalization strategies in the context of ontological mapping and 3D visualization.

The main goal is to comparatively analyze normalization methods across multiple levels and demonstrate their implementation in network databases. The research results serve to ensure the semantic consistency of network data and improve the processes of its representation in a three-dimensional visual environment.

Related works

The issue of data normalization has emerged in recent years as one of the most pressing areas of concern in working with large volumes of data. Researchers have proposed various approaches to this issue, with the aim of simplifying the process of combining and analyzing data with different formats, units, and semantics by bringing them into a single form. Below is an analysis of the main scientific approaches and developments in data normalization.

Kalayani Sankpal and K. V. In the Metre article entitled "An Overview of Data Normalization Methods" [1], various levels, methods of data normalization, and their limitations are described in detail. The authors view data normalization as a three-step process: normalization performed at the record, field, and value levels. In this approach, the goal is not only to remove duplicate records, but also to ensure complete semantic consistency of information from different sources. Normalization is divided into three levels depending on the data structure:

Record normalization is the process of identifying and unifying records that belong to the same object but are expressed differently. Algorithms for detecting and merging duplicates are often used for this purpose. Culotta and his colleagues [2] called this process "canonization" and proposed a "distance editing" method to determine the similarity between records. In their study, the central (canonical) record is created by finding the closest match between records. However, since this approach did not take into account differences in value levels, the normalized data was not completely accurate.

The Swoosh system [3] similarly treats data as connections between objects. It aligns and merges records using entity resolution algorithms. However, since this approach did not take into account differences in value levels, the normalized data was not completely accurate. Meanwhile, the approach proposed by Yongquan Dong and Dragut[4] developed a three-step model for record level normalization : merging records, comparing field values, and creating a resulting single record. This model was published in the IEEE Transactions on Knowledge and Data Engineering journal, where high accuracy in the automatic selection of similar records was achieved.

Field level normalization is the process of converting the values of each column in data records to the same format. In this approach, formatting, aligning units of measurement, and reducing text differences become increasingly important. For example, "TSEU" and "Tashkent State Economic University" mean the same thing but are written differently. Vic and his colleagues [5] proposed a working model based on schema matching and coreference resolution to address this issue. They performed attribute matching using a (discriminative) model that distinguishes those that do not correspond to a reference. This approach was effective in analyzing structural similarities between data, but did not fully ensure semantic correspondence at the level of meaning. The "object normalization" system [6] proposed by Tejada, Knoblock and Minton also used string ratings and user confidence ratings (confidentiality ratings) to compare attributes. This system performed well in normalizing data from web sources, but in many cases required user intervention. In addition, the label normalization method developed by Dragut and Mann [7] proposes matching data by assigning meaningful labels (semantic labels) to attributes. This approach is considered important in ontological systems, especially for improving the semantic consistency of data.

Value level normalization. Normalization at the value level aims to ensure semantic and linguistic consistency of attribute values at the most granular level of data. At this stage, technologies such as text similarity recognition, abbreviation and synonym normalization, and NLP (natural language processing) are often used.

(cc) BY

The model proposed by Sanpal and Meter [1] uses ranking algorithms such as frequency ranking, length ranking, centroid ranking, and similarity analysis based on bigram and edit distance for normalization at the value level. They use the mining abbreviation–definition pairs algorithm to identify abbreviations and the template collocation method to search for text references. The advantage of this approach is that it not only eliminates duplication at the record level, but also takes into account the semantic proximity between values. However, the fact that natural language processing (NLP) methods were not used in this system limited its accuracy. The researchers noted that much higher accuracy could be achieved by introducing an NLP-based model at later stages.

Methodology:multi-level normalization framework

The normalization framework adopted in this study operates across three hierarchical levels as established in [1] and is applied to each of twelve data types identified ing network database schemas:

Table 1.1.

Normalization levels and their objectives

Type of normalisation	Main task	Goal	Example
Record-level	Identifying and merging records that represent the same object but are stored differently	Removing duplicates, storing data in a unified format	"Aliyev A." and "A. Aliyev" – merged as the same person
Field-level	Bringing attribute or column values in records to a uniform format	Eliminating inconsistencies in format, consistency, and written form	"TSEU" va "Tashkent state economical university" – standardized the same value
Value-level	Linguistic and semantic adaptation of each value	Eliminating synonyms, abbreviations, and spelling differences, creating semantic harmony	"prof.","profesor" or "Profesor" – are rewritten the only form

Network databases contain attributes in a wide variety of formats. Table 1.2 presents the taxonomy of twelve data types used in this study, along with representative examples and common usage contexts.

Table 1.2.

Data types

N	Type of data	Description	Example	Usage
1.	**Numeric/ Decimal/Float**	Represents whole or decimal numbers	45, 3.14, -120.56	Price, measurement, percentage, calculated values
2.	**Integer/ SmallInt/BigInt**	Stores natural or whole numbers, without fractional parts.	1, 0, 2025, -56	Identification numbers, quantity, amount
3.	**Text / Char / Varchar / String**	Letters, words, texts, or alphanumeric characters.	"Ziyoda", "Tashkent", "ATU-2025"	Name, location, text, code, description

(cc) BY

4.	**Enumerated / Categorical / Enum**	Selects one of a limited number of categories.	("Male", "Female"), ("Active", "Inactive")	Status, tour, category
5.	**Date / Time / Timestamp / Datetime**	Stores dates, times, or time intervals.	2025-11-03, 12:30:00	Logs, date of birth, time of process
6.	**Boolean / Bit**	True/false binary values	True, False, 1, 0	Conditions, circumstances, activity indicator
7.	**Blob / Binary / Image / JSON**	Non-text objects: image, video, document, or structured information. Stores natural or whole numbers, without fractional parts.	01001011, JSON fayl, .jpg, .pdf	Multimedia, configuration, file storage
8.	**Spatial / Geometry / Point**	Represents geographical coordinates and shapes	(41.3111, 69.2797)	Location, GIS, network topology
9.	**Money / Currency**	Financial values in currency units	15000.00, $200	Accounting, transactions, economic analysis
10.	**Array / Set / List**	Multi-values attributes	{1, 2, 3} or {"A", "B", "C"}	Multi-valued attributes, storing multi-valued attributes
11.	**JSON / XML / Object**	Stores semi-structured, hierarchical or nested data objects	{"name":"Ziyoda","age":25}	Name, place, text, code, description
12.	**NULL**	Absent or unknown value	NULL	Missing data of any type

Now, based on this table, normalization methods are selected individually, first using the steps presented in Table 1.1, in accordance with each data type and its task, and then analyzing the appropriate normalization methods for each data type.

1. Numerical data normalization is mainly achieved through scaling. The most commonly used model is the min-max normalization method[8].

$$x_i' = \frac{x_i - min(x)}{max(x) - min(x)}$$

here:

- $x_i \ - \ real\ value$
- Min(x) and max(x) – range of values
- $x_i' - result\ of\ normalisation$

An alternative model can also be used – the Z-score normalization method:

$$z_i = \frac{x_i - \mu}{\sigma}$$

here:

$\mu \ - \ mean, \sigma \ - \ standard\ deviation$

Normalization of the Z-score results in a mean value of 0 and a variance of 1.

2. Integer data

Integers are usually discrete values that are normalized by reindexing or interval coding[9]:

$$C_j = \begin{cases} -1, & if\ a_i < x_i < b_i \\ 0, & else \end{cases}$$

$here: a_i, b_i \ - \ interval\ boundaries, C_j \ - \ symbol\ of\ category$

3. Text data:

Semantic normalization is performed on text data. Mathematically, this is expressed through a vectorization model[10]:

(cc) BY

$$w_i' = \frac{w_i}{\|w\|}$$

here: w_i – word frequency, $\|w\|$ – length of vector. The lemmatization is as follows:

$$f: \text{word form} \rightarrow \text{root form}$$

4. Categorical data:

Normalization of categorical data means representing it in digital space.

The One-hot encoding model looks like this [11]:

$$x_i' = [x_{i1}, x_{i2}, \ldots, x_{in}]$$

5. Time data normalization is performed in two stages depending on the data format and value:

a) normalization by format:

$$t_{norm} = f_{iso}(t_{input})$$

Here f_{iso} – a function that converts dates from different formats to the ISO 8691 format.

b) Value model:

$$t' = \frac{t - t_{min}}{t_{max} - t_{min}}$$

Here:

- t_{min} – earliest time
- t_{min} – latest time
- t' – normalized value (between 0-1)

in which the time interval is selected for all time-related data[12].

6. Logical data. Logical values are the simplest type of data in the binary system, whose values are always as follows:

$$x \in \{True, False\} \quad yoki \quad x \in \{1, 0\}$$

Therefore, there is no need to normalize this type of data, but logical values may differ across different sources[13].

Table 1.3.

Boolean value representations across sources

Source	Representation	Format type
DB 1	True/false	Logical
DB 2	1/0	Numeric
DB 3	Yes/no	Textual
DB 4	On/off	Linguistic
DB 5	Active/Inactive	Categorical

The normalization mapping is: {True, 1, "On", "Yes", "Active"} ⇒ 1; {False, 0, "Off", "No", "Inactive"} ⇒ 0.

7. Binary/File data[14]:

$$y = \frac{1}{\alpha} ln(1 + (\alpha x))$$

Here x – the pixel brightness in the image, $\alpha > 0$ – appropriate transformation parameter, and then fitted to a Gaussian distribution as follows:

$$z_i = \frac{y_i - \mu_y}{\sigma_y}$$

8. Spatial data. Adapted by scaling spatial coordinates[15]:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}, \qquad y' = \frac{y - y_{min}}{y_{max} - y_{min}},$$

For example, location in Tashkent: (41.31, 69.28)

Global range: x= [0,90], y= [0,180], $x' = 0.458$, $y' = 0.385$

9. Monetary data. The unit exchange model is used in currency normalization

$$V' = V \times R$$

Therefore, the range of values is reduced with the log scaling method[16]*:*

$$x' = log_{10}(x)$$

10. Array/Set data. Array attributes may contain null values, duplicates, or irregular sequences. Normalization proceeds in three steps

- *Empty values are excluded:* $S' = \{s_i \in S \mid s_i \neq NULL\}$
- *Duplicate values are removed:* $S'' = distinct(S')$
- *The arrangement is carried out:* $S''' = \{ s_i \in S'' \mid s_1 \leq s_2 \ldots \leq s_n$

11. Data in JSON and XML formats will consist of nested structures. When working with such structures, it is necessary that they appear flat, i.e., each internal object is represented as a separate attribute. This process ensures ease of integration and analysis of semi-structured data.

$$(K, V) \Rightarrow \{K_1: V_1, K_2: V_2, \ldots, K_n: V_n\}$$

12. Missing values. One of the simplest ways to fill in missing values is to fill them with the mean value. In this approach, the mean value of the observed values is replaced with the missing value in each attribute. [18]:

$$x_i = \begin{cases} x_i, & if\ x_i\ is\ observed \\ \frac{1}{n}\sum_{j=1}^{n} x_j, & if\ x_i\ is\ missing \end{cases}$$

Experimental results

A synthetic database of 1,000 records was constructed in Python to evaluate the

(cc) BY

normalization framework. The dataset included attributes spanning all twelve data types: numeric (age, salary, rating), text (full_name, comment, status), categorical (gender), boolean (is_active), temporal (join_date, deleted_at), binary (profile_image), JSON (metadata), and a file path attribute.

Before normalization the data consisted of values in different ranges, and the attributes did not have the same scale. For example, in the *salary* column, values ranged from 2,000 to 9,000, while in the text columns, a mixture of uppercase and lowercase letters was used. Keys in JSON objects are not sorted, and in some records, the *deleted_at* column had NULL values.

Below is a table of fragments obtained randomly using the *sample()* function from the table state before normalization :

index	full_name	age	salary	gender	is_active	join_date	profile_image	metadata	comment	rating	file_path	status	deleted_at
413	Gulbahor Sharipova	46	6818.42	Female	false	2025-03-08	R3VsYmFob3IgU2hhcmlwb3ZhX3Byb2ZpbGU=	{"department": "IT", "position": "Engineer", "experience_years": 2}	New staff	1.92	/files/Gulbahor_Sharipova.pdf	Inactive	NaT
484	Madina Yusupova	25	6691.4	Male	true	2023-02-16	TWFkaW5hIFl1c3Vwb3ZhX3Byb2ZpbGU=	{"department": "IT", "position": "Developer", "experience_years": 10}	Active employee	4.5	/files/Madina_Yusupova.pdf	Inactive	2025-03-14 12:58:09.676037
552	Nigora Rustamova	50	7612.02	Female	true	2021-07-11	Tmlnb3JhIFJ1c3RhbW92YV9wcm9maWxl	{"department": "Marketing", "position": "Lecturer", "experience_years": 5}	Resigned	4.42	/files/Nigora_Rustamova.pdf	Inactive	NaT

Fig.1. Before normalization table

(cc) BY *dtai.uz*

The following steps were taken during the normalization process:

1. Numerical attributes (age, salary, rating) were scaled to the range [0,1] using the Min–max scaling method.
2. Text attributes (full_name, gender, comment, Status) were converted to lowercase.
3. JSON object keys were sorted alphabetically and converted to a semantically consistent form.
4. Empty values (null) are filled with the word "none".

The following steps were performed during normalization.

index	full_name	age	salary	gender	is_active	join_date	profile_image	metadata	comment	rating	file_path	status	deleted_at
35	malika karimova	0.095238	0.124541	female	False	2023-12-30	TWFsaWthIEthcmltb3ZhX3Byb2ZpbGU=	{"department": "HR", "experience_years": 10, "...	active employee	0.634085	/files/Malika_Karimova.pdf	inactive	2025-01-04 19:12:51.271402
123	boburxon ergashev	0.738095	0.059815	male	False	2022-01-22	Qm9idXJ4b24gRXJnYXNoZXZfcHJvZmlsZQ==	{"department": "IT", "experience_years": 8, "p...	active employee	0.786967	/files/Boburxon_Ergashev.pdf	inactive	2024-07-14 04:21:42.470415
122	ziyoda norqulova	0.690476	0.306059	female	True	2023-08-12	Wml5b2RhIE5vcnF1bG92YV9wcm9maWxl	{"department": "HR", "experience_years": 5, "p...	active employee	0.934837	/files/Ziyoda_Norqulova.pdf	active	2024-07-04 19:25:22.042599

Fig.2. Normalized dataset sample

Analysis of the results showed that:

- The values of numerical attributes were reduced to the same scale, which made it possible to compare them during analysis and visualization.
- The difference between uppercase and lowercase letters in text fields disappeared, and the inscriptions became uniform.
- Since the JSON objects were sorted, the semi-structured data acquired semantic consistency. Analysis of the results showed that:

(cc) BY

- The values of numerical attributes were reduced to the same scale, which made it possible to compare them during analysis and visualization.
- The difference between uppercase and lowercase letters in text fields disappeared, and the inscriptions became uniform.
- Since JSON objects were sorted, semi-structured data acquired semantic consistency.
- Adding the character "none" instead of empty values ensured data completeness.
- Overall, as a result of normalization, the database acquired structural, numerical, and semantic harmony. The dispersion of numerical attributes decreased by an average of 70 percent, while text and JSON type adaptations ensured 100 percent consistency. This approach created a ready-made dataset for the next stage-data integration, ontological mapping, and 3D visualization.

Discussion

The results demonstrate that a type-aware, multi-level normalization approach substantially improves data quality across all dimensions measured. Several findings merit discussion.

First, the choice of numeric normalization method has material consequences for downstream processing. Min-max scaling, applied in this study, is well-suited for bounded visualization tasks but would be suboptimal if the salary or age attributes contained extreme outliers. In such cases, Z-score standardization or robust scaling (using the interquartile range) would be preferable. The comparative framework in Section 4.1 provides explicit criteria for this decision.

Second, the multi-level framing — record, field, value — proves useful for organizing normalization tasks. In the experimental dataset, record-level normalization (deduplication) was not required due to the synthetic nature of the data. However, in real-world network databases, record-level normalization would typically precede field and value normalization, and its omission would undermine the consistency achieved at lower levels.

Third, the treatment of JSON and semi-structured data as a distinct type — rather than as a special case of text — is justified by the results. Alphabetical key sorting achieved 100% structural consistency, enabling downstream processes to treat metadata as a reliable, ordered attribute rather than an opaque string.

Fourth, boolean unification, while conceptually simple, is non-trivial in practice: the same logical state may be represented in at least five distinct surface forms across different database systems (Table 5). Automated detection and mapping of these forms requires a predefined vocabulary, which this framework provides.

The primary limitation of this study is the use of a synthetic dataset. Real network databases introduce additional complications: inconsistent null semantics, multilingual text, heterogeneous coordinate systems, and currency fluctuation. Future work will validate the framework on real-world network databases, investigate NLP-enhanced value-level normalization, and examine the interaction between normalization choices and the quality of ontological mappings and 3D visualizations.

Conclusion

This study presents a comparative analysis of multi-level normalization techniques and their practical implementation in network databases. Experiments with a synthetically structured database of 1,000 records made it possible to develop a unified mechanism for processing various types of data - numeric, text, JSON, binary, and time values. Through normalization, all values were reduced to a single range, and structural and semantic consistency in the data structure was ensured. Numeric columns were scaled from 0 to 1, text data was converted to the same format, JSON objects were sorted, and empty values were filled in. As a result, the data was fully ready for analysis and visualization. Through normalization, all values were reduced to a single range, ensuring structural and semantic consistency.

Literature

1. Sankpal, K. A., & Metre, K. V. (2020). *A review on data normalization techniques*. *International Journal of Engineering Research & Technology (IJERT)*, 9(6), 885–889.

(cc) BY

https://doi.org/10.17577/IJERTV9IS060915
.

2. Culotta, A., Mccallum, A., & Betz, J. (2007). *Canonicalization of database records using adaptive edit distance*. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)* (pp. 104–113). ACM Press. https://doi.org/10.1145/1281192.1281207.
3. Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E., & Widom, J. (2009). *Swoosh: A generic approach to entity resolution*. *The VLDB Journal*, 18(1), 255-276. https://doi.org/10.1007/s00778-008-0098-x.
4. Dong, Y., Dragut, E. C., & Meng, W. (2019). *Normalization of duplicate records from multiple sources*. *IEEE Transactions on Knowledge and Data Engineering*, 31(4), 769-782. https://doi.org/10.1109/TKDE.2018.2875227.
5. Wick, M. L., Rohanimanesh, K., Schultz, K., & McCallum, A. (2008). *A unified aproach for schema matching, coreference and canonicalization*. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)* (pp. 722-730). ACM. https://doi.org/10.1145/1401890.1401977
6. Tejada, S., Knoblock, C. A., & Minton, S. (2001). *Learning object identification rules for information integration. Information Systems*, 26(8), 607-633. https://doi.org/10.1016/S0306-4379(01)00033-4
7. Dragut, E. C., & Meng, W. (2006). *Meaningful labeling of integrated query interfaces.* In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06)* (pp. 679-690). VLDB Endowment.
8. Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann.
9. Pang-Ning Tan et al. (2019). *Introduction to Data Mining.* Pearson Education.
10. Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
11. Bishop, C. M. *Pattern Recognition and Machine Learning.* Springer.
12. Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann Publishers
13. Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill.
14. Ballé, J., Laparra, V., & Simoncelli, E. P. (2015). *Density modeling of images using a generalized normalization transformation*.
15. Shekhar, S., Chawla, S., Zhang, P., & Lazarz, A. (2015). *Spatial Databases: A Tour* (2nd ed.). Pearson Education.
16. Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann Publishers.
17. Abadi, D. J. (2018). The Design and Implementation of Modern Column-Oriented Database Systems. Springer.
18. Little, R. J. A., & Rubin, D. B. (2019). *Statistical Analysis with Missing Data* (3rd ed.). Wiley.

(cc) BY *dtai.uz*