Notes on Intuitionistic Fuzzy Sets
Print ISSN 1310–4926, Online ISSN 2367–8283
2026, Volume 32, Number 1, 30–39
DOI: 10.7546/nifs.2026.32.1.30-39

A modifying operator for temporal intuitionistic fuzzy index matrices with an application to autonomous self-managing databases

Veselina Bureva [1] **and Krassimir Atanassov** [2]

[1] Laboratory of Intelligent Systems, Burgas State University "Prof. Dr. Assen Zlatarov"
1 "Prof. Yakimov" Blvd., Burgas 8010, Bulgaria
e-mail: `veselina-bureva@uniburgas.bg`

[2] Department of Bioinformatics and Mathematical Modelling,
Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 105, Sofia 1113, Bulgaria
e-mails: `krat@bas.bg, k.t.atanassov@gmail.com`

Received: 28 November 2025 **Revised:** 20 January 2026
Accepted: 23 February 2026 **Online First:** 10 March 2026

Abstract: In the current investigation, a modifying operator over temporal intuitionistic fuzzy index matrices (TIFIM) is presented. Thereafter, an example of autonomous priority-based self-managing database system using intuitionistic fuzzy logic is developed. The modifying operator is implemented for priorities updating. The priorities are changed according to three cases. In the first case, the modifying operator increases the degree of membership and decreases the degree of non-membership if the record is frequently searched. In the second case, the modifying operator decreases the degree of membership and increases the degree of non-membership if the record is rarely searched. In the third case, the modifying operator set the record priority to $\langle 0,1 \rangle$ and deleted it the record is not searched more than n days.

Keywords: Autonomous self-managing database, Dynamic priorities, Temporal intuitionistic fuzzy index matrix.
2020 Mathematics Subject Classification: 03E72.

1 Brief remarks on temporal intuitionistic fuzzy index matrices

The theory of index matrices (IMs) is described in [5]. IMs have assigned indices on the rows and on the columns. The values can be real numbers, logical values, intuitionistic fuzzy pairs. Intuitionistic fuzzy evaluations are based on the intuitionistic fuzzy sets and can be presented in the form of intuitionistic fuzzy pairs [7]. The theory of IFS, operations and operators are described in [3, 4]. Therefore, the types of index matrices extensions include intuitionistic fuzzy index matrices (IFIM), *n*-dimensional intuitionistic fuzzy index matrices (*n*-DIFIM), Index matrices with function-type elements (IMFE), temporal intuitionistic fuzzy index matrices (TIFIM). In the current investigation, we will use TIFIMs. Each TIFIM has three dimensions assigned with letters *K*, *L* and *T*, where *K* and *L* are sets of indices and *T* is some fixed temporal scale. The TIFIM has the following form:

$$A(T) = \left[K, L, T, \langle \mu_{k_i,l_j,\tau}, \nu_{k_i,l_j,\tau}\rangle\right]$$

$$\equiv \left\{ \begin{array}{c|ccccc} \tau & l_1 & \dots & l_j & \dots & l_n \\ \hline k_1 & \langle \mu_{k_1,l_1,\tau}, \nu_{k_1,l_1,\tau}\rangle & \dots & \langle \mu_{k_1,l_j,\tau}, \nu_{k_1,l_j,\tau}\rangle & \dots & \langle \mu_{k_1,l_n,\tau}, \nu_{k_1,l_n,\tau}\rangle \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ k_i & \langle \mu_{k_i,l_1,\tau}, \nu_{k_i,l_1,\tau}\rangle & \dots & \langle \mu_{k_i,l_j,\tau}, \nu_{k_i,l_j,\tau}\rangle & \dots & \langle \mu_{k_i,l_n,\tau}, \nu_{k_i,l_n,\tau}\rangle \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ k_m & \langle \mu_{k_m,l_1,\tau}, \nu_{k_m,l_1,\tau}\rangle & \dots & \langle \mu_{k_m,l_j,\tau}, \nu_{k_m,l_j,\tau}\rangle & \dots & \langle \mu_{k_m,l_n,\tau}, \nu_{k_m,l_n,\tau}\rangle \end{array} \middle| \tau \in T \right\}$$

where τ is an element of *T*, i.e., a time-moment for every $\tau \in T$,$1 \le i \le m, 1 \le j \le n$:

$$\mu_{k_i,l_j,\tau},\ \nu_{k_i,l_j,\tau},\ \mu_{k_i,l_j,\tau} + \nu_{k_i,l_j,\tau} \in [0,1].$$

Let us have the time-moments τ_1 and τ_2 such that $\{\tau_1, \tau_2\} \in T$ and let us have the constants $c, d \in [0,1]$ such that $c + d \le 1$. Therefore, we define the following modifying operator for TIFIM:

$$M(A(T), c, d) = \left[K,\ L,\ T,\ \langle \rho_{k_i,l_j,\tau}, \sigma_{k_i,l_j,\tau}\rangle\right],$$

where

$$\langle \rho_{k_i,l_j,\tau}, \sigma_{k_i,l_j,\tau}\rangle =$$

$$= \begin{cases} \langle \mu_{k_i,l_j,\tau} + c.\dfrac{\tau_1 - \tau}{\tau_1}.\ \nu_{k_i,l_j,\tau},\ d.\dfrac{\tau}{\tau_1}.\ \nu_{k_i,l_j,\tau}\rangle, & \text{if } k_i \in K, l_j \in L, \tau \in T \text{ and } \tau \le \tau_1 \\ \langle \dfrac{\tau_2 - \tau}{\tau_2 - \tau_1}.\mu_{k_i,l_j,\tau}, d.\dfrac{\tau - \tau_1}{\tau_2 - \tau_1}.\ \nu_{k_i,l_j,\tau}\rangle, & \text{if } k_i \in K, l_j \in L, \tau \in T \text{ and } \tau_1 < \tau < \tau_2 \\ \langle 0,1\rangle, & \text{if } k_i \in K, l_j \in L, \tau \in T \text{ and } \tau \ge \tau_2 \end{cases}$$

We can check that $\rho_{k_i,l_j,\tau}, \sigma_{k_i,l_j,\tau}, \rho_{k_i,l_j,\tau} + \sigma_{k_i,l_j,\tau} \in [0, 1]$. A geometric interpretation is shown in Figure 1.

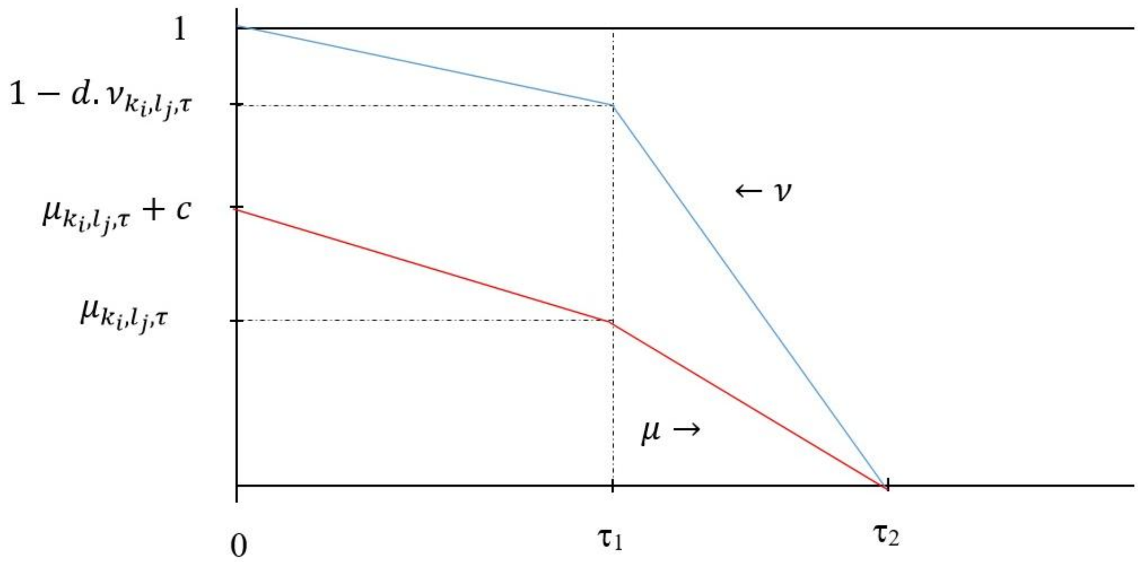
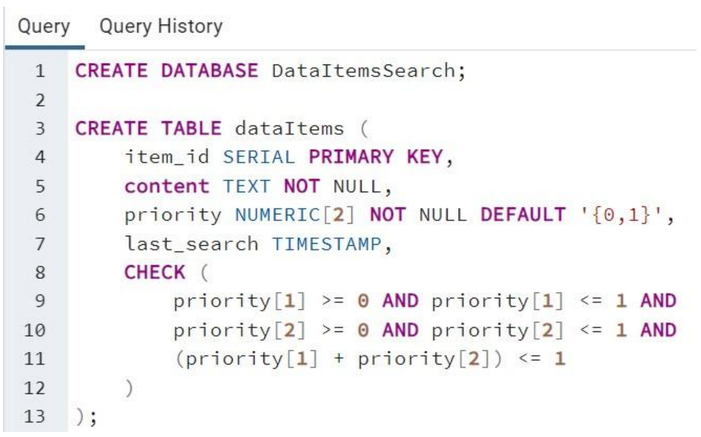


Figure 1. Geometric interpretation

In the current investigation, the searching data frequency is observed. The datasets priorities are calculated. Depending on the appropriate time-moments and the searching frequency, the database states are updated. The frequency estimation is defined using the notation of intuitionistic fuzzy sets. Similar investigation for intuitionistic fuzzy evaluations calculation while performing INSERT and UPDATE statements are discussed in [9]. Different approached for self-managing systems are also investigated [1, 8, 11, 12, 14–17]. Generalized net models of expert systems are constructed and explained in series of papers [2, 6, 10, 13]. In the next section, knowledge from the theory of expert systems, the new modifying operator and databases are integrated to develop an example for a self-management system.

2 Autonomous priority-based self-managing database system

The presented investigation describes an autonomous priority-based self-managing database system. The system dynamically adjusts the priority of stored records based on their search frequency. It automatically reduces the priority of rarely accessed data, deletes outdated information, and reinforces the importance of frequently selected records. Through scheduled maintenance and adaptive control functions, the database maintains optimal relevance and efficiency without human intervention. The records priorities are presented in the form of intuitionistic fuzzy pairs. Therefore, an example of modifying operator over TIFIM is presented. The TIFIM is represented as a data table. The database is implemented using the *PostgreSQL* database management system. An array data type is used for the "*priority*" column. The *dataItems* table's SQL code is presented in Figure 2. An additional rule to the "*priority*" column is assigned: its values and their sum have to be in the interval [0, 1]. The column *last_search* contains the information for the latest data search.

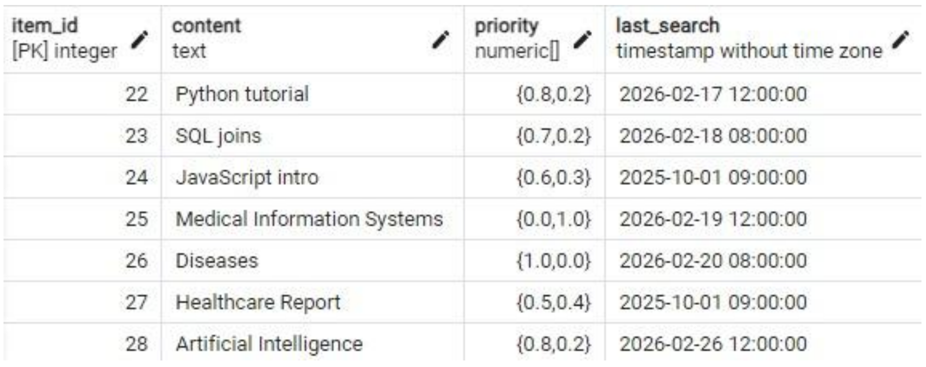


```sql
CREATE DATABASE DataItemsSearch;

CREATE TABLE dataItems (
    item_id SERIAL PRIMARY KEY,
    content TEXT NOT NULL,
    priority NUMERIC[2] NOT NULL DEFAULT '{0,1}',
    last_search TIMESTAMP,
    CHECK (
        priority[1] >= 0 AND priority[1] <= 1 AND
        priority[2] >= 0 AND priority[2] <= 1 AND
        (priority[1] + priority[2]) <= 1
    )
);
```

Figure 2. SQL statement for *dataItems* table

The records into the *dataItems* table are inserted. The priority field contains the intuitionistic fuzzy pairs. They are calculated according to the records searching frequency. In each time moment, the table has a different state depending on the priority field. The values are calculated dynamically. The data table is presented in Figure 3.



item_id [PK] integer	content text	priority numeric[]	last_search timestamp without time zone
22	Python tutorial	{0.8,0.2}	2026-02-17 12:00:00
23	SQL joins	{0.7,0.2}	2026-02-18 08:00:00
24	JavaScript intro	{0.6,0.3}	2025-10-01 09:00:00
25	Medical Information Systems	{0.0,1.0}	2026-02-19 12:00:00
26	Diseases	{1.0,0.0}	2026-02-20 08:00:00
27	Healthcare Report	{0.5,0.4}	2025-10-01 09:00:00
28	Artificial Intelligence	{0.8,0.2}	2026-02-26 12:00:00

Figure 3. State of the *dataItems* table for time moment *t*

The first step of the autonomous priority-based self-managing database system development is to provide the methodology for priority calculation. It has the following form:

- If the record is frequently searched, the degree of membership increases and the degree of non-membership decreases;
- If the record is infrequently, rarely searched (between x and y days), the degree of membership decreases and the degree of non-membership increases;
- If the record is non searched more than n days then the row is deleted after setting its priority to $\langle 0,1 \rangle$

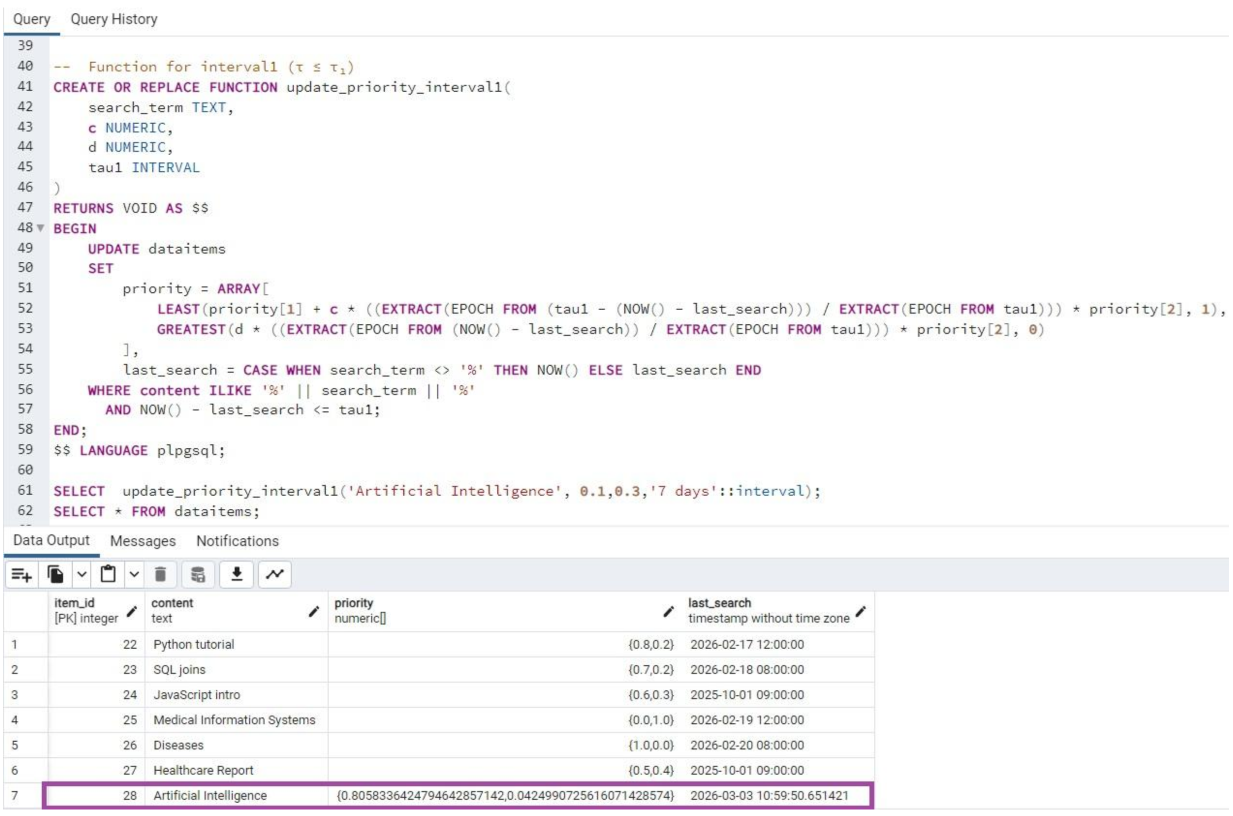
Therefore, the next logic step is to define four functions:

- *update_priority_on_interval1*() – increase the priority when the user searches in the row;
- *update_priority_on_interval2*() – decrease priority for inactive records;
- *update_priority_on_interval3*()– setting values $\langle 0, 1 \rangle$ to old records that are not searched n days.
- *delete_old_data*() – delete old records that are not searched n days.

The *update_priority_on_interval1*() function created tracks the data search and changes the priority of the selected records. Thereafter, the function have to be automated. It has to be activated of each SELECT statement execution. The searched text, the constants *c* and *d* and time interval have to be assigned in the function-calling step. They are used for degrees of membership and non-membership calculation. In the presented example a word "*Artificial Intelligence*" searching is performed. The value 0.1 and 0.3 are assigned to the constants *c* and *d*. After the function execution the priority of the record with *id* = 7 and content "*Artificial Intelligence*" is updated (Figure 4).

In the second step the *update_priority_on_interval2*() function is created. It provides functionalities for decreasing the priority of rarely searched items. In the presented example, the time interval between 7 and 30 days is considered for the rare searched items determination. Therefore, the records that satisfy the condition are considered as infrequently searched. Their priorities are decreased using the constants *c* and *d* respectively for degrees of membership and non-membership. In the presented example, the priority of the records with content not searched between 7 and 30 days is modified (Figure 5).

In the third step *update_priority_on_interval3*() function is created. It has the capabilities for setting the values ⟨0, 1⟩ to the records that are not searched more than 30 days. In the function, the searched text, the constants *c* and *d* and time interval have to be assigned. In the presented example four records are modified (Figure 6).

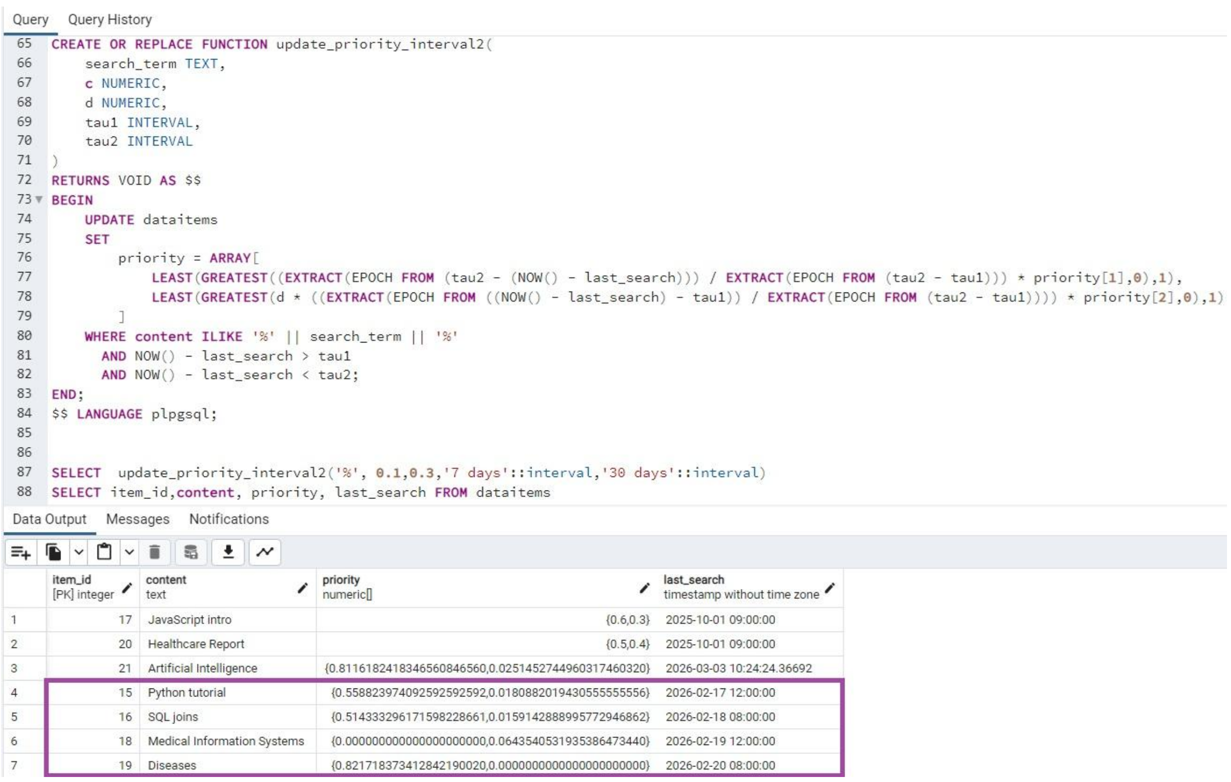


```
--  Function for interval1 (τ ≤ τ₁)
CREATE OR REPLACE FUNCTION update_priority_interval1(
    search_term TEXT,
    c NUMERIC,
    d NUMERIC,
    tau1 INTERVAL
)
RETURNS VOID AS $$
BEGIN
    UPDATE dataitems
    SET
        priority = ARRAY[
            LEAST(priority[1] + c * ((EXTRACT(EPOCH FROM (tau1 - (NOW() - last_search))) / EXTRACT(EPOCH FROM tau1))) * priority[2], 1),
            GREATEST(d * ((EXTRACT(EPOCH FROM (NOW() - last_search)) / EXTRACT(EPOCH FROM tau1))) * priority[2], 0)
        ],
        last_search = CASE WHEN search_term <> '%' THEN NOW() ELSE last_search END
    WHERE content ILIKE '%' || search_term || '%'
      AND NOW() - last_search <= tau1;
END;
$$ LANGUAGE plpgsql;

SELECT  update_priority_interval1('Artificial Intelligence', 0.1,0.3,'7 days'::interval);
SELECT * FROM dataitems;
```

	item_id [PK] integer	content text	priority numeric[]	last_search timestamp without time zone
1	22	Python tutorial	{0.8,0.2}	2026-02-17 12:00:00
2	23	SQL joins	{0.7,0.2}	2026-02-18 08:00:00
3	24	JavaScript intro	{0.6,0.3}	2025-10-01 09:00:00
4	25	Medical Information Systems	{0.0,1.0}	2026-02-19 12:00:00
5	26	Diseases	{1.0,0.0}	2026-02-20 08:00:00
6	27	Healthcare Report	{0.5,0.4}	2025-10-01 09:00:00
7	28	Artificial Intelligence	{0.8058336424794642857142,0.0424990725616071428574}	2026-03-03 10:59:50.651421

Figure 4. Function that changes the priority of the recording according to the search activities

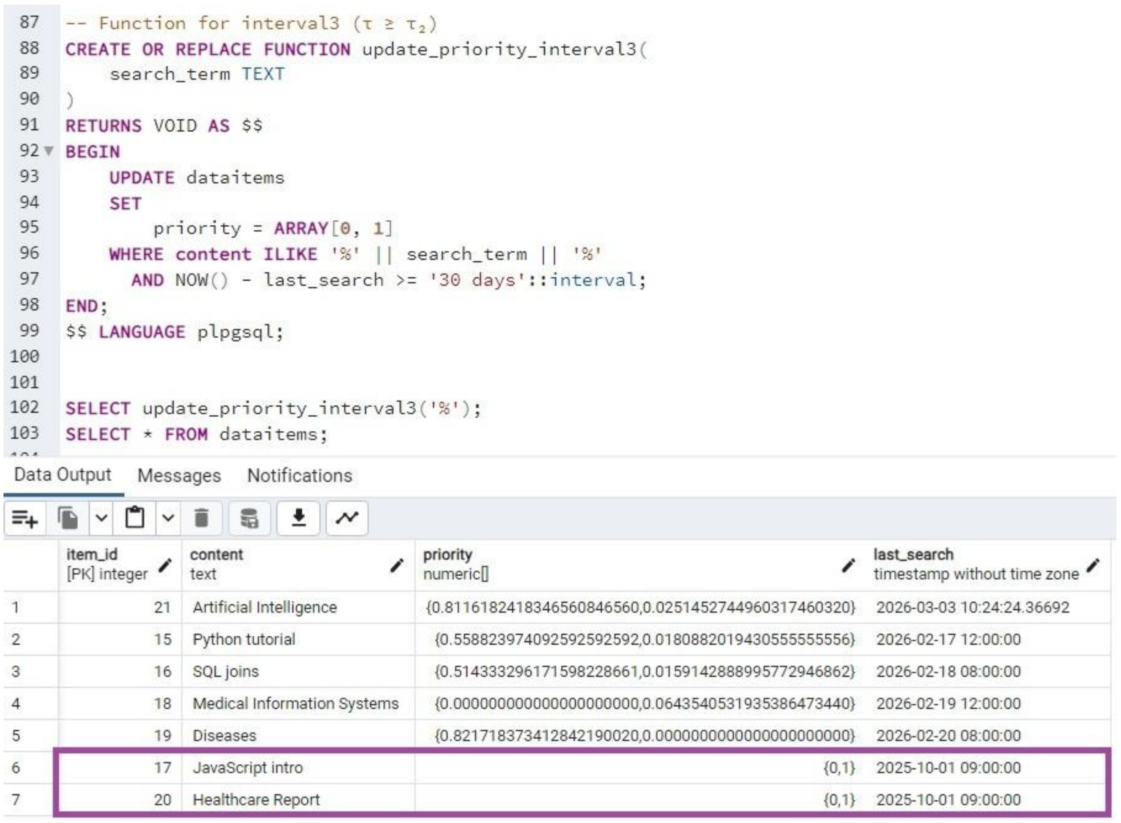


```sql
CREATE OR REPLACE FUNCTION update_priority_interval2(
    search_term TEXT,
    c NUMERIC,
    d NUMERIC,
    tau1 INTERVAL,
    tau2 INTERVAL
)
RETURNS VOID AS $$
BEGIN
    UPDATE dataitems
    SET
        priority = ARRAY[
            LEAST(GREATEST((EXTRACT(EPOCH FROM (tau2 - (NOW() - last_search))) / EXTRACT(EPOCH FROM (tau2 - tau1))) * priority[1],0),1),
            LEAST(GREATEST(d * ((EXTRACT(EPOCH FROM ((NOW() - last_search) - tau1)) / EXTRACT(EPOCH FROM (tau2 - tau1)))) * priority[2],0),1)
        ]
    WHERE content ILIKE '%' || search_term || '%'
      AND NOW() - last_search > tau1
      AND NOW() - last_search < tau2;
END;
$$ LANGUAGE plpgsql;


SELECT  update_priority_interval2('%', 0.1,0.3,'7 days'::interval,'30 days'::interval)
SELECT item_id,content, priority, last_search FROM dataitems
```

	item_id [PK] integer	content text	priority numeric[]	last_search timestamp without time zone
1	17	JavaScript intro	{0.6,0.3}	2025-10-01 09:00:00
2	20	Healthcare Report	{0.5,0.4}	2025-10-01 09:00:00
3	21	Artificial Intelligence	{0.8116182418346560846560,0.0251452744960317460320}	2026-03-03 10:24:24.36692
4	15	Python tutorial	{0.558823974092592592592,0.0180882019430555555556}	2026-02-17 12:00:00
5	16	SQL joins	{0.514333296171598228661,0.0159142888995772946862}	2026-02-18 08:00:00
6	18	Medical Information Systems	{0.000000000000000000000,0.0643540531935386473440}	2026-02-19 12:00:00
7	19	Diseases	{0.821718373412842190020,0.0000000000000000000000}	2026-02-20 08:00:00

Figure 5. Function that decrease the priority of the rarely searched items



```sql
-- Function for interval3 (τ ≥ τ₂)
CREATE OR REPLACE FUNCTION update_priority_interval3(
    search_term TEXT
)
RETURNS VOID AS $$
BEGIN
    UPDATE dataitems
    SET
        priority = ARRAY[0, 1]
    WHERE content ILIKE '%' || search_term || '%'
      AND NOW() - last_search >= '30 days'::interval;
END;
$$ LANGUAGE plpgsql;


SELECT update_priority_interval3('%');
SELECT * FROM dataitems;
```

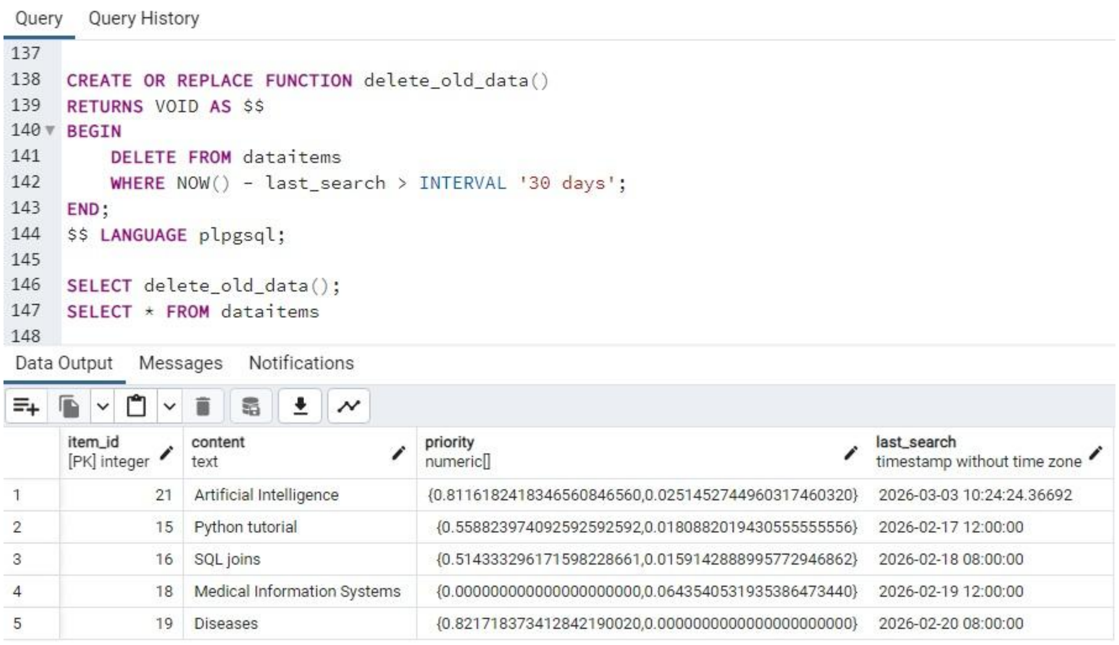
	item_id [PK] integer	content text	priority numeric[]	last_search timestamp without time zone
1	21	Artificial Intelligence	{0.8116182418346560846560,0.0251452744960317460320}	2026-03-03 10:24:24.36692
2	15	Python tutorial	{0.558823974092592592592,0.0180882019430555555556}	2026-02-17 12:00:00
3	16	SQL joins	{0.514333296171598228661,0.0159142888995772946862}	2026-02-18 08:00:00
4	18	Medical Information Systems	{0.000000000000000000000,0.0643540531935386473440}	2026-02-19 12:00:00
5	19	Diseases	{0.821718373412842190020,0.0000000000000000000000}	2026-02-20 08:00:00
6	17	JavaScript intro	{0,1}	2025-10-01 09:00:00
7	20	Healthcare Report	{0,1}	2025-10-01 09:00:00

Figure 6. Function that deletes the non-searched items

The result of the *delete_old_data*() function execution is presented in Figure 7. The records that are not searched more than a month are deleted from the *dataItems* table. The new stat of the table contains the frequently searched data according to the previously defined conditions.

In the next step, the authors present the step of the priorities updating automation. The system is set to be self-managing. The *update_priority_on_interval1*() is rewritten as trigger function *trg_update_priority_on_search*() that enables autonomous self-management in the database by capturing every user search request. Upon insertion of a new search query into search_log, the function activates the priority recalculation algorithm. Therefore, the system will continuously adapt item priorities based on real-time user behavior, without manual intervention (Figure 8).

The tasks of decreasing priority and deleing records are atomized using *pg_cron* extension for PostgreSQL. Firstly, the extension has to be installed. Thereafter three jobs are created. The first job is used for modifying priorities of the not searched data between 7 and 30 days. The update will be every night at 2:00 pm. The second job is activated every night at 2.05 pm. The third job will delete the rarely searched data (>30 days) every night at 3:00 pm. The users can select the schedule of the defined jobs (Figure 9).

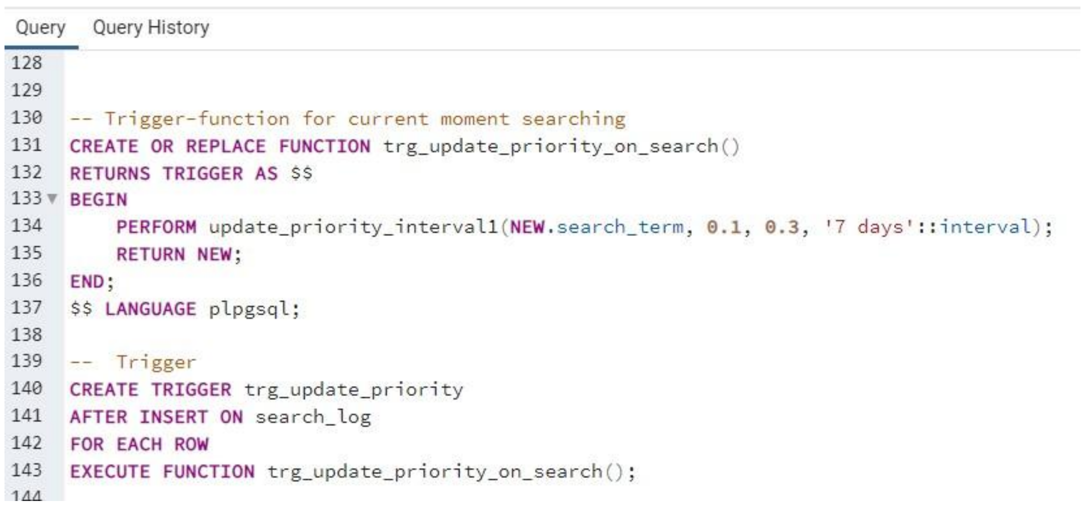


```
CREATE OR REPLACE FUNCTION delete_old_data()
RETURNS VOID AS $$
BEGIN
    DELETE FROM dataitems
    WHERE NOW() - last_search > INTERVAL '30 days';
END;
$$ LANGUAGE plpgsql;

SELECT delete_old_data();
SELECT * FROM dataitems
```

	item_id [PK] integer	content text	priority numeric[]	last_search timestamp without time zone
1	21	Artificial Intelligence	{0.8116182418346560846560,0.0251452744960317460320}	2026-03-03 10:24:24.36692
2	15	Python tutorial	{0.558823974092592592592,0.0180882019430555555556}	2026-02-17 12:00:00
3	16	SQL joins	{0.514333296171598228661,0.0159142888995772946862}	2026-02-18 08:00:00
4	18	Medical Information Systems	{0.000000000000000000000,0.0643540531935386473440}	2026-02-19 12:00:00
5	19	Diseases	{0.821718373412842190020,0.0000000000000000000000}	2026-02-20 08:00:00

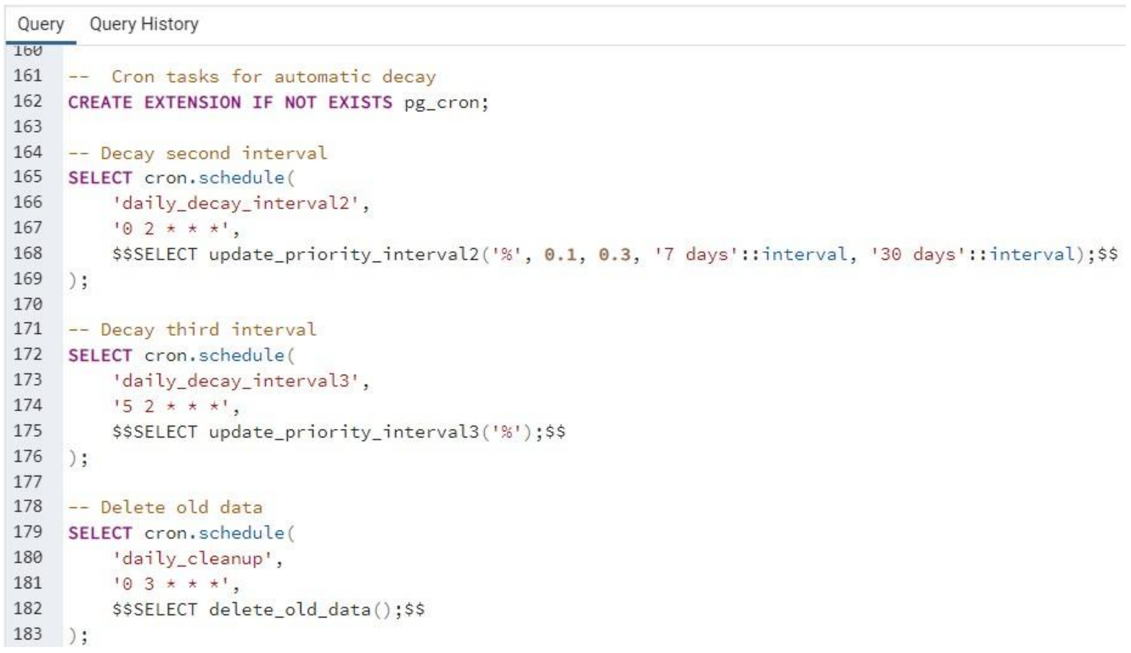
Figure 7. Result of the delete function execution



```
-- Trigger-function for current moment searching
CREATE OR REPLACE FUNCTION trg_update_priority_on_search()
RETURNS TRIGGER AS $$
BEGIN
    PERFORM update_priority_interval1(NEW.search_term, 0.1, 0.3, '7 days'::interval);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

--  Trigger
CREATE TRIGGER trg_update_priority
AFTER INSERT ON search_log
FOR EACH ROW
EXECUTE FUNCTION trg_update_priority_on_search();
```

Figure 8. Trigger function *trg_update_priority_on_search*()



```
--  Cron tasks for automatic decay
CREATE EXTENSION IF NOT EXISTS pg_cron;

-- Decay second interval
SELECT cron.schedule(
    'daily_decay_interval2',
    '0 2 * * *',
    $$SELECT update_priority_interval2('%', 0.1, 0.3, '7 days'::interval, '30 days'::interval);$$
);

-- Decay third interval
SELECT cron.schedule(
    'daily_decay_interval3',
    '5 2 * * *',
    $$SELECT update_priority_interval3('%');$$
);

-- Delete old data
SELECT cron.schedule(
    'daily_cleanup',
    '0 3 * * *',
    $$SELECT delete_old_data();$$
);
```

Figure 9. Cron jobs

3 Conclusion

In the current investigation a modifying operator over temporal intuitionistic fuzzy index matrices is discussed. An application in the field of self-management databases is presented. The dynamic priorities are implemented. The activities related to the modifying operator are automated in the autonomous database system.

Acknowledgements

This research was funded by the Bulgarian National Science Fund under the Grant KP-06-N72/8 from 14.12.2023 “Intuitionistic fuzzy methods for data analysis with an emphasis on the blood donation system in Bulgaria”.

References

[1] Abbasi, M., Bernardo, M. V., Váz, P., Silva, J., & Martins, P. (2024). Adaptive and scalable database management with machine learning integration: A PostgreSQL case study. *Information*, 15(9), Article 574.

[2] Atanassov, K. (1998). *Generalized Nets in Artificial Intelligence. Volume 1: Generalized Nets and Expert Systems*. “Prof. Marin Drinov” Publishing House of the Bulgarian Academy of Sciences, Sofia.

[3] Atanassov, K. (1999). *Intuitionistic Fuzzy Sets: Theory and Applications*. Springer, Heidelberg.

[4] Atanassov, K. (2012). *On Intuitionistic Fuzzy Sets Theory*. Springer, Berlin.

[5] Atanassov, K. (2014). *Index Matrices: Towards an Augmented Matrix Calculus*. Studies in Computational Intelligence Series, Vol. 573, Springer, Cham.

[6] Atanassov, K. (2020). A generalized net model of an intuitionistic fuzzy expert system. *Notes on Intuitionistic Fuzzy Sets*, 26(1), 46–68.

[7] Atanassov, K., Szmidt, E., & Kacprzyk, J. (2013). On intuitionistic fuzzy pairs. *Notes on Intuitionistic Fuzzy Sets*, 19(3), 1–13.

[8] Avula, S. B. (2025). Oracle autonomous database for transforming database administration by harnessing artificial intelligence. In: *Global Conference in Emerging Technology*, Pune, India, 2025, 1–7, doi: 10.1109/GINOTECH63460.2025.11076743.

[9] Bureva, V., Petrov, P., Andonov, V., & Atanassov, K. (2023). Intuitionistic fuzzy evaluation of user requests frequency. In: Atanassov, K. T., *et al.* (Eds.) *Uncertainty and Imprecision in Decision Making and Decision Support - New Advances, Challenges, and Perspectives*. IWIFSGN BOS/SOR 2022. Lecture Notes in Networks and Systems, Vol. 793, pp. 15 – 21. Springer, Cham.

[10] Chountas, P., Atanassov, K., Sotirova, E., & Bureva, V. (2016). Generalized net model of an expert system dealing with temporal hypothesis. In: Andreasen, T., Christiansen, H., Kacprzyk, J., Larsen, H., Pasi, G., Pivert, O., De Tré, G., Vila, M. A., Yazici, A., Zadrożny, S. (Eds.) *Flexible Query Answering Systems 2015*. Advances in Intelligent Systems and Computing, Vol. 400, 473–481. Springer, Cham.

[11] Helskyaho, H., Yu, J., & Yu, K. (2021). Oracle autonomous database for machine learning. In: *Machine Learning for Oracle Database Professionals*, pp. 97–133. Apress, Berkeley, CA.

[12] Kavuluri, H., Avula, S. B., & Sirimalla, A. (2025). Performance tuning for cloud-based databases Oracle/Postgres: Analyzing query optimization techniques. *Journal of Wireless Mobile Networks*, 16(2), 459–475.

[13] Kolev, B., El-Darzi, E., Sotirova, E., Petrounias, I., Atanassov, K., Chountas, P., & Kodogiannis, V. (2006). *Generalized Nets in Artificial Intelligence. Volume 3: Generalized Nets, Relation Databases and Expert Systems*. “Prof. Marin Drinov” Publishing House of the Bulgarian Academy of Sciences, Sofia.

[14] Kossmann, J., & Schlosser, R. (2019). A framework for self-managing database systems. In: *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*, Macao, China, pp. 100–106.

[15] Oloruntoba, O. (2025). AI-Driven autonomous database management: Self-tuning, predictive query optimization, and intelligent indexing in enterprise IT environments. *World Journal of Advanced Research and Reviews*, 25(02), 1558–1580.

[16] Sharma, B M., Krishnakumar, K .M. & Panda, R. (2022). *Oracle Autonomous Database in Enterprise Architecture: Utilize Oracle Cloud Infrastructure Autonomous Databases for better consolidation, automation, and security*. Packt Publishing, Birmingham.

[17] Sheikhkhoshkar, M., El-Haouzi, H., Aubry, A., Hamzeh, F., & Rahimian, F. (2025). A data-driven and knowledge-based decision support system for optimized construction planning and control. *Automation in Construction*, 173, Article ID 106066.