IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/46ICT/2025/4647325/
Vol. 2025 (35), Article ID 4647325

Design And Application of an Administrative Panel for Managing the Employees Activities of a Research and Teaching Unit: The Case of Employee Evaluation System at Czestochowa University of Technology, Poland*

Szymon BERSKI and Kamila PASTERNAK

Czestochowa University of Technology, Faculty of Computer Science and Artificial Intelligence, Czestochowa, Poland, Czestochowa, Poland,

Correspondence should be addressed to: Szymon BERSKI, szymon.berski@pcz.pl

* Presented at the 46th IBIMA International Conference, 26-27 November 2025, Ronda, Spain

Abstract

The article presents the structure, organisation and examples of operation of the university employee activity administrator panel, which is the main element of the employee evaluation system (EES) at Czestochowa University of Technology (CUT). EES consists of a database server based on PostgreSQL and LibreOffice Base (LO Base) software, which is part of the LibreOffice suite. The pgAdmin4 v.9.2 for Chrome Browser database management software was used to test individual solutions at the design and validation stage, which enabled the design and creation of employee evaluation form templates, as well as the use of the macro language to build advanced queries in forms and work with LO Base subforms. This paper extends this system to include a panel for administering employee activity (AEP) and also describes modifications to the database schema on which the EES system is based. Moreover, the usefulness of the created system in the management and administration environment of higher education institutions was demonstrated in the area of employee evaluation and human resources management as well as in the area of organisational security in terms of continuity of access to the system, data security and software servicing.

Keywords: PostgreSQL, LibreOffice Basic scripts, open-source software, organisational security

Introduction

In various areas of the economy and administration, many different types of programmes and applications are used, i.e., to create websites, documents and reports. Many of those tools require a licence fee. However, it is worth noting that there is much open-source software which also can be used in this area. It should be emphasised that low or no costs are not the only advantage of using such solutions. These types of tools allow for the creation of flexible solutions by enabling modification and customisation of code to specific and unique requirements. When considering the advantages of open-source software, security improvements also deserve special attention. It can be argued that open-source projects are subject to constant scrutiny due to their access to a large developer community. It can be found in literature (The Rising (2024)) that the quality of code from open-source projects exceeds that of proprietary tools.

One of the most popular open-source tools in the office suite category is LibreOffice described in the Libre Guide (2025). It is a powerful set of office tools for word processing, spreadsheets, presentations, databases, and more. From this article's point of view, the most important feature of LibreOffice is the ability to assign your own scripts to menu items, icons, dialogue controls and events. Such scripts are also called macros. A macro is a sequence of stored commands or keystrokes to automate repetitive tasks. Such functionality allows complex procedures to be performed with a single click or keyboard shortcut, saving time and eliminating the need to manually repeat the same actions.

Cite this Article as: Szymon BERSKI and Kamila PASTERNAK, Vol. 2025 (35) "Design and Application of an Administrative Panel for Managing the Employees Activities of a Research and Teaching Unit: The Case of Employee Evaluation System at Czestochowa University of Technology, Poland" Communications of International Proceedings, Vol. 2025 (35), Article ID 4647325, https://doi.org/10.5171/2025.4647325

An important aspect of macros is the way in which they are stored. Fig. 1 presents an exemplary hierarchical structure of macro organisation in LibreOffice. Modules can group macros, libraries can group modules, and library containers can group libraries.

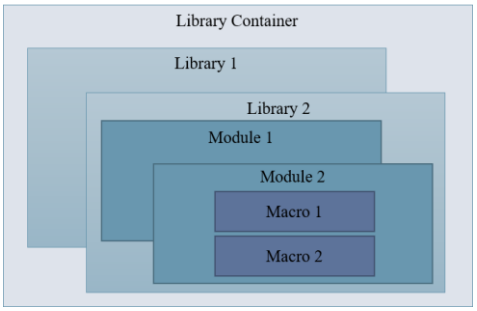


Fig 1. The hierarchical structure of macro libraries in LibreOffice

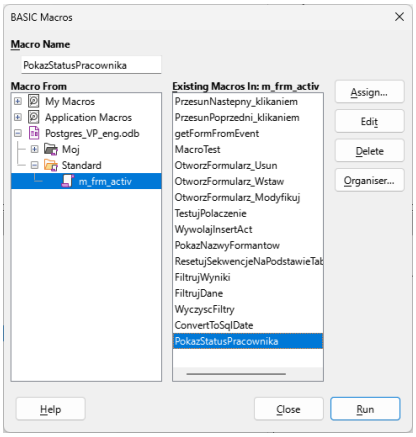


Fig 2. EES basic Macro dialog

The application itself functions as two library containers – macros distributed with LibreOffice (LibreOffice Macros) and macros created by the user (My Macros). Every LibreOffice document also is a library container. If a macro is required for a specific document, it seems to be useful to store that macro in the document. However, it should be emphasised that a macro contained in a document is visible only to that document. Storing a macro in a document is good practice when the document will be shared (the macro is included with the document). If the macro should be available to all documents, it is better to store it in My Macros container. A library container can contain one or more libraries, and a library can contain one or more modules. Similar functionality is usually grouped within modules. The lowest-level objects are macros, which are stored in modules. The hierarchical structure described above can be seen in Fig. 2. It shows the Basic Macros dialogue, which can be used to create new macros and organise libraries. More information about macro organisation in LibreOffice can be found, e.g., in LibreOffice Guide (2020).

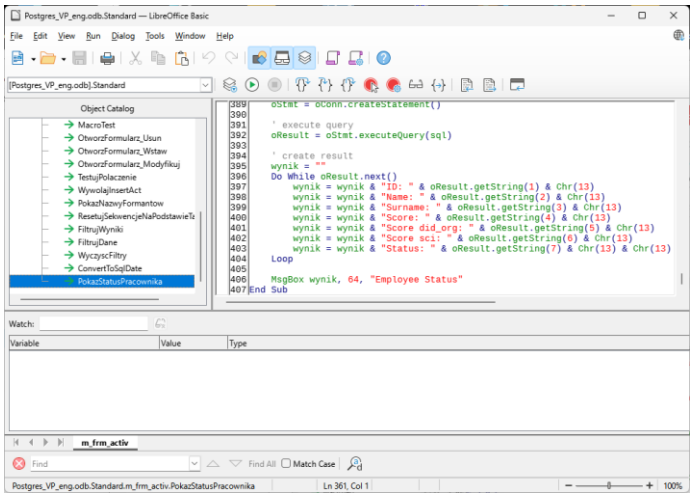


Fig 3. Macros manager in LO Base

It should be mentioned that LibreOffice supports the following scripting languages: Basic, JavaScript, Python, and BeanShell (Pitonyak, A. D. (2004)). Moreover, developers can use high-level languages, e.g. Java, C++ to externally control LibreOffice. API documentation can be found in API documentation (2025). Due to preparing the administrative units of the analysed university for the LibreOffice office package software migration, the work macros written in LO Basic language are used in this study. The tool used to manage the macros in LO Base is presented in Fig. 3.

This paper includes an extension of an IT system model for employee evaluation (EES), which was presented in Berski S. et al. (2023) and expanded in Berski S. et al. (2025). These papers used LibreOffice in combination with the PostgreSQL database as a system for managing the data necessary to correctly complete the required documents. Literature demonstrates this approach. For instance, this approach was utilised by Golomingi R. et al. (2025). This combination was used to build a database and user interface for the analysis of forensic medical data. Klaasse J. at al. (2021) proposed to use the PostgreSQL database and LibreOffice to store forensic data obtained from experts from various disciplines. The developed system, based on information retrieved from the database, allows for automatic completion of necessary forms and the evaluation of employees according to criteria defined in the applicable regulations. The described system is based on the example of employee evaluation which is used in Czestochowa University of Technology (CUT). The main principles of this evaluation are included in Rector's Ordinance (2022), which is presented in detail in Berski S. et al. (2023) and Berski S. et al. (2025). The basis of preparing an employee evaluation were d1, d2 and d3 documents, which contain all the criteria, procedures and forms necessary for the periodic evaluation of academic teachers. d1 and d2 documents were the basis for forms created using the developed model of the evaluation system. More details about d1, d2 and d3 documents can be found in Berski S. et al. (2023) and Berski S. et al. (2025).

The aim and the research methodology

The aim of this work is to develop, build, and test an activity administrator module for the evaluation process of university staff at a research and teaching unit (AEP). Considering the security of the software being developed, the recommendations in this area, both global (Inter (2010) and Tridgell, J. (2025)) and local from university authorities, as well as the limited costs of this project, the authors decided to utilise available office suite solutions and a fully open-source database management system. The module under analysis is a crucial part of the EES system described above, which was also built using fully open-source solutions.

The module implemented at the research and development unit will fulfil the following tasks:

- it will be used by administrative staff to manage university employee activities for the EES purposes,
- it will be used by management staff to control and manage human resources and work organization within the university's organizational units,
- It will also be a key element in increasing the security of the unit's organization (CUT) by: leveraging institutional resources (CUT), simplifying the management of employee activities, becoming independent of commercial office software licenses, and increasing employee security in terms of work planning and organization, which is embedded in the rector's new strategy.

The need to build an additional module for EES emerged during testing of form d2 and d3 documents, which fulfilled their role in generating ready-made reports but turned out to be too detailed for managing employee activities.

Moreover, there has been a change in the legal order at the university because, from 2024, a new Rector's Ordinance (2024) regarding the employee evaluation process was published. This document extends the evaluation period compared to the previous one. It was decided to rebuild the previous logical database schema because such changes can occur more frequently. The logical schema of the database has also been changed to include the ability to evaluate part-time employees. The full text of the Rector's Ordinance (in Polish) can be found in Rector's Ordinance (2024).

Methodology of research and Results

The updated model of the EES is based on the client-server architecture. The model, which contains the administrative panel for managing employee evaluation activities (AEP), was based on the open-source PostgreSQL RDBMS and the LibreOffice (LO) office suite, which includes an application for LO Base databases and a macros manager. The versions of software are presented in Table 1 and were updated in comparison to the previous EES model presented in Berski S. et al. (2025).

Table 1: Version of open-source software used for AEP analysis

Software	Version

LibreOffice Writer and Base installed in Windows 11 OS	LibreOffice Community 25.2.7.2 (X86_64)
PostgreSQL installed in Linux Ubuntu Server	PostgreSQL 14.18 (Ubuntu 14.18-0ubuntu0.22.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0, 64-bit

The *valid_pers* database schema, which underlies the entire EES system and the newly designed AEP module, was restored using the standard *pg_dump* database backup software. No anomalies, errors, or data inconsistencies were detected during the restoration of the database schema to the new server version.

Due to changes in the above-mentioned Rector's Ordinance (2024) and the possibility of employing part-time employees, for the proper functioning of EES it was necessary to change the software logic, i.e., the logical structure of the database, in terms of changing the evaluation period and criteria related to part-time positions. For this purpose, from the *employee* relation (Fig. 4), the *working_time* attribute, which in the previous schema was only an attribute of the relation, was constructed as a foreign key with a reference to a separate relation called *working_time* (Fig. 6). Support for a more complex database schema will require the use of macros to handle buttons in the administration panel form (a separate entity extended with additional properties necessary to correct the evaluation criterion for part-time employees). The use of macros to handle form buttons enabled the use of array types for years, the number of which is calculated using the COUNT function built into the database in accordance with the criteria improved in relation to the previous model, presented in Table 2.

Table 2: New criteria for different employee positions according to new regulations (Rector's Ordinance (2024))

Position type of employee	**Period of time [time interval]**	**Evaluation results [points for positive assessment]**		
		Scientific activities in publication	**other activities (scientific, teaching and organisation)**	**Total**
research-teaching	*n* years	$\geq n \cdot 60$		$\geq n \cdot 100$
research		$\geq n \cdot 80$		
teaching (didactic)			$\geq n \cdot 80$	

Furthermore, the new logical schema includes an additional *ewaluacja* relation (all evaluations or individual selected ones can be analysed by assigning them a separate identifier, not just the date in the activity-related tables). These changes can be observed in the ERD diagrams presented in Figs 4 and 5.

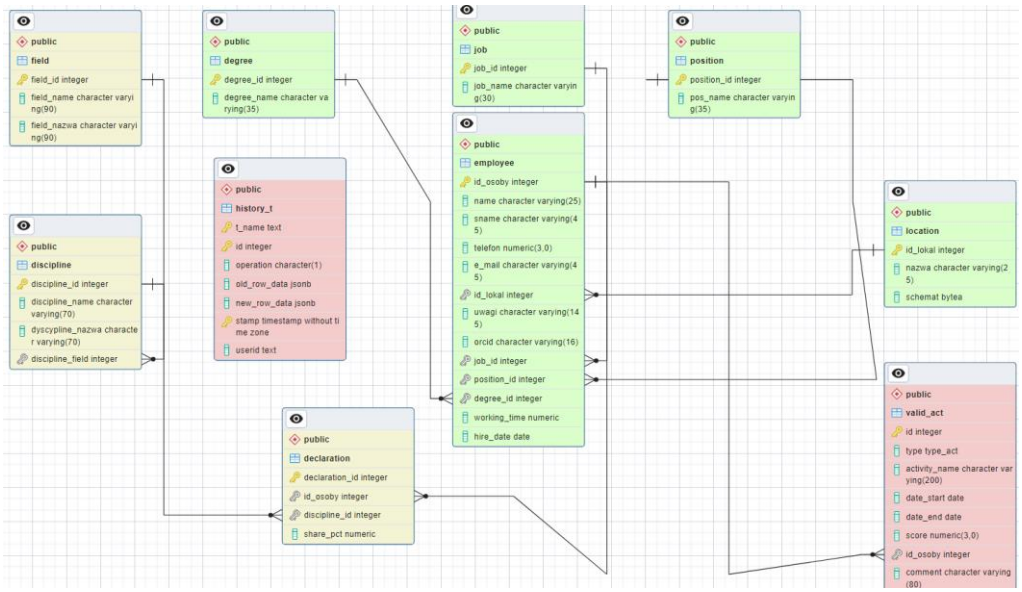


Fig 4. ERD for EES database schema before changes according to Berski S. et al. (2025)

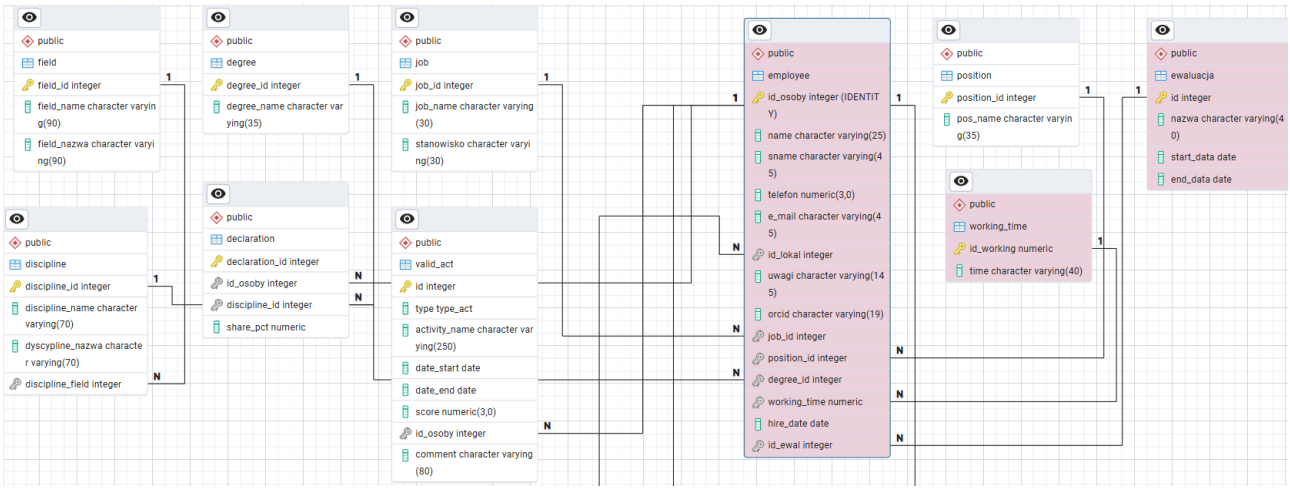
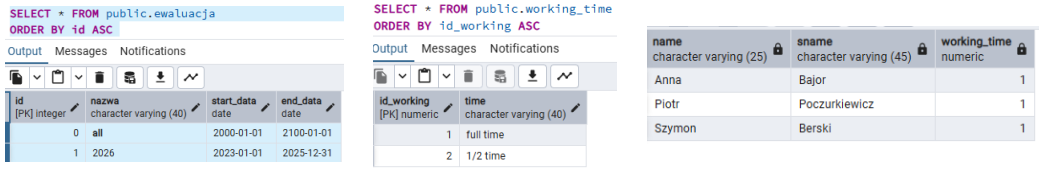


Fig 5. ERD database schema for AEP module

The data stored in the newly created *working_time* and *ewaluacja* relations are presented in Fig. 6.



```
SELECT * FROM public.ewaluacja
ORDER BY id ASC
```

Output | Messages | Notifications

id [PK] integer	nazwa character varying (40)	start_data date	end_data date
0	all	2000-01-01	2100-01-01
1	2026	2023-01-01	2025-12-31

a)

```
SELECT * FROM public.working_time
ORDER BY id_working ASC
```

Output | Messages | Notifications

id_working [PK] numeric	time character varying (40)
1	full time
2	1/2 time

b)

name character varying (25)	sname character varying (45)	working_time numeric
Anna	Bajor	1
Piotr	Poczurkiewicz	1
Szymon	Berski	1

c)

Fig 6. Example data for new relations

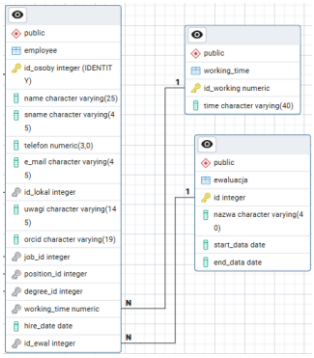
The structure of these objects and the data stored in them can be used, for example, to modify functions or views in the database in order to correct the evaluation criterion (Fig.6a), or for additional analyses related to the identification of individual evaluations (Fig. 6b). Additionally, Fig. 6c also shows how to display data for full-time employees. Personal data were modified for presentation purposes in this publication.

Fig. 7a shows an example of correcting the evaluation of individual criteria in the *employee_score()* function body. And in Fig. 7b is shown a fragment of the ERD for new relations and their connections with the *employee* table.

```
CREATE OR REPLACE FUNCTION public.employee_score(
    id_employeea integer,
    id_working numeric,
    years integer[])
    RETURNS TABLE(id_osoby integer, imie character varying,
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
    ROWS 1000

AS $BODY$
DECLARE
    no_of_years INTEGER := cardinality(years);
    working_time NUMERIC := 1;
    total_crit INTEGER := 100 * no_of_years * working_time;
    sci_crit INTEGER := 60 * no_of_years * working_time;
BEGIN
    RETURN QUERY
    SELECT
```

a)



b)

Fig 7. Example corrections in database function (a) and relationship between new tables (b)

This solution guarantees that the database's logical structure is resistant to future changes made in the length of the employee's evaluation period and also differentiates the evaluation criteria depending on the employee's job type.

The graphical design of AEP was created as a form with nested subforms activated by buttons running macros written in LO Basic using the techniques described in Berski S. et al. (2023) and Berski S. et al. (2025). The arrangement of individual operations is presented using buttons in Fig. 8.

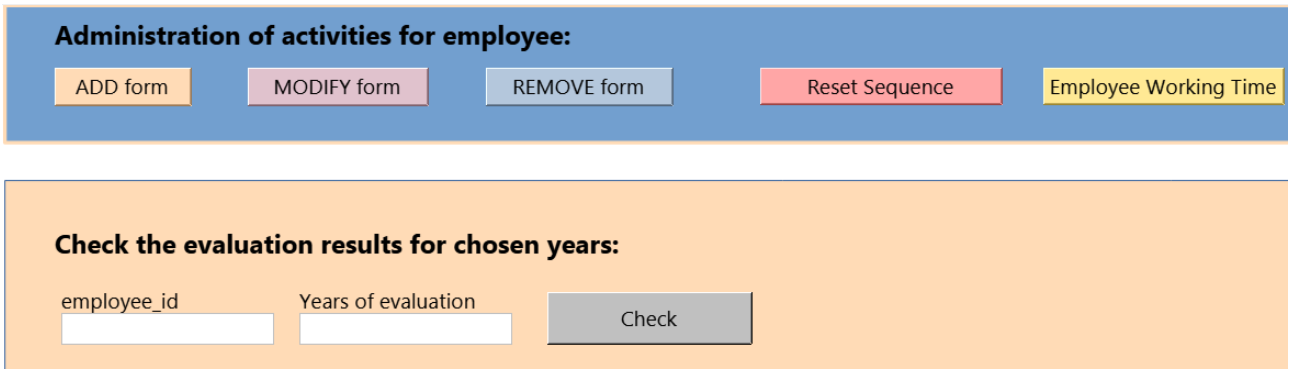


Fig 8. The administration panel (AEP) layout

The AEP building layer was constructed as a main form (*panel_valid_activity*) using the LO Base form manager presented in Fig. 9 together with subforms: *activity_adding_form*, *activity_modification_form* and *activity_removing_form*.

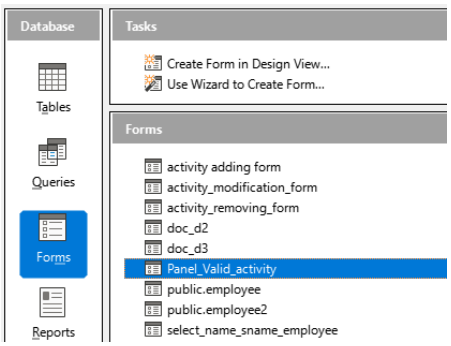


Fig 9. LO Base forms collection for building AEP

The structure and description of individual elements of the administrator panel are presented for the main operations performed by the end user, i.e., adding, modifying, and deleting activities. Queries on views and other objects in the PostgreSQL database system were used to verify the database status. Screenshots of the verification results and the contents of individual forms and the macros that trigger them are shown in the figures below.

To add an activity with specific content for a selected employee, the *ADD form* button should be pressed, which invokes the *activity_adding_form* subform shown in Fig. 10a. After correctly filling out the fields and pressing the button, a macro linked to the form is launched, the code for which is shown in Fig. 10b. The *id_activity* field is not specified on the *activity_addition_form* because it is the primary key automatically generated by the appropriate *valid_act_seq* sequence in the database. Verification of the new record insertion into the database was performed using standard SQL queries in *pgAdmin 4* software, and the results of this query are included in Fig. 10c.

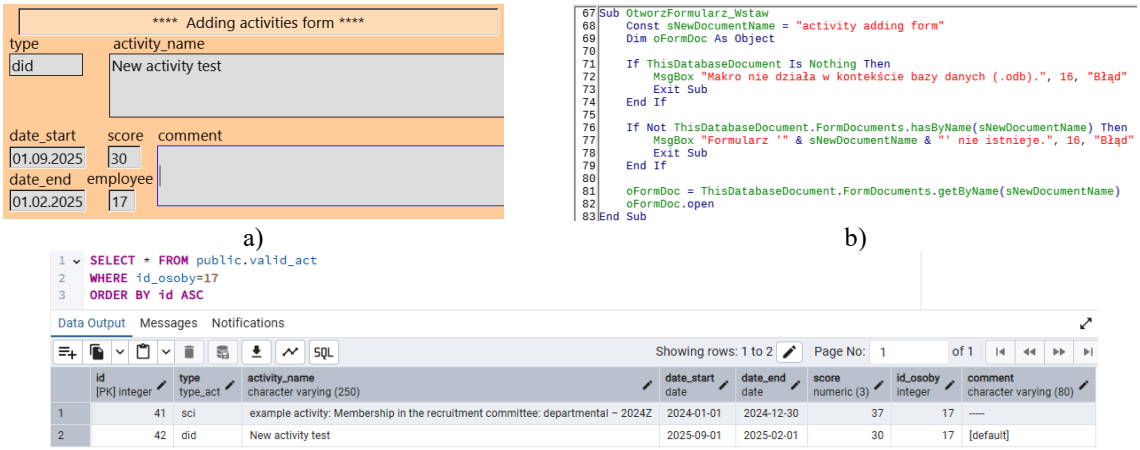


Fig 10. Activity adding form layer (a), macro (b) and SQL query verifying new rows in database (c)

To modify an activity with a specific identifier (key in the *id_activity* column), which can be searched for using criteria such as its name, type, date, and the employee performing the activity, click the *MODIFY form* button, which invokes the *activity_modification_form* subform shown in Fig. 11a. After updating the selected fields outside the primary key and using the records panel, a macro linked to the form will be run, the code of which is shown in Fig. 11b. The *id_activity* field is locked for editing on the form to maintain data consistency in the database; however, the sequence index can be rebuilt, but only using the *Reset Sequence* button on the AEP form. The modifications to the selected record were also verified using standard SQL queries in *pgAdmin 4* software, and the results of this query are shown in Fig. 11c.

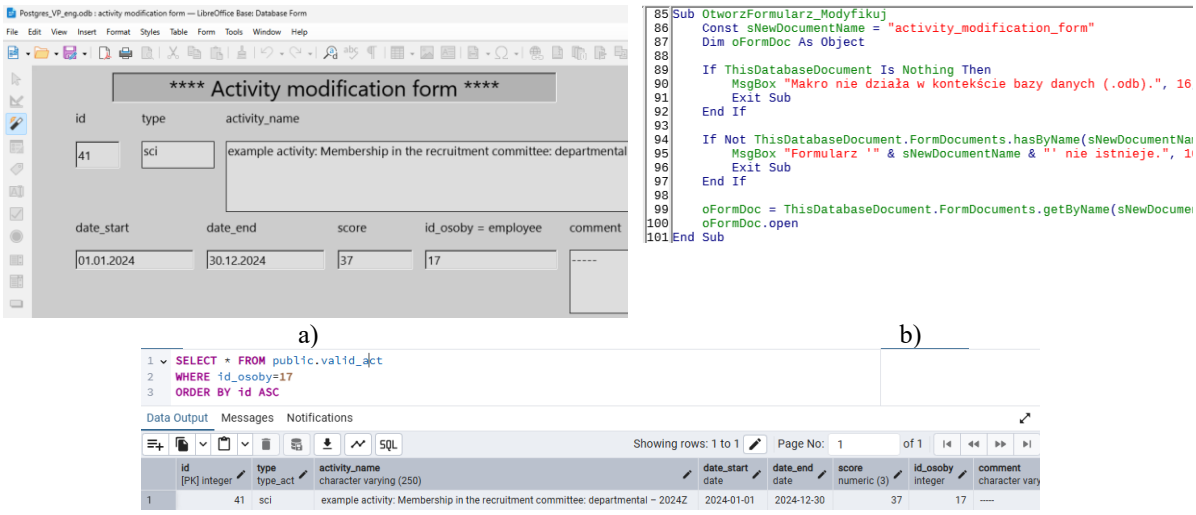


Fig 11. Activity modification form layer (a), macro (b) and SQL query verifying modifications in database (c)

It should be noted that during execution of the form, all integrity constraints in the database are checked, and their violations are signalled with appropriate messages, e.g., an attempt to change the record identifier shown in Fig. 12.

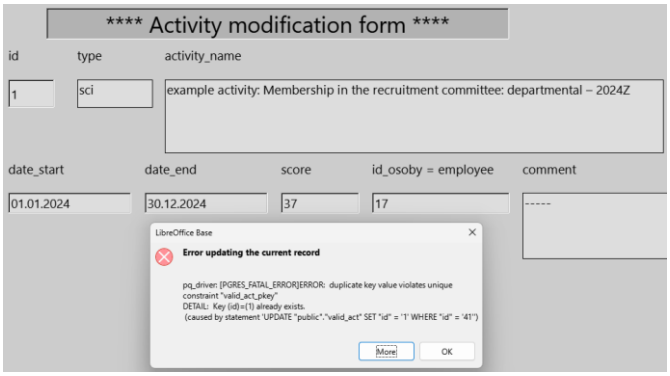
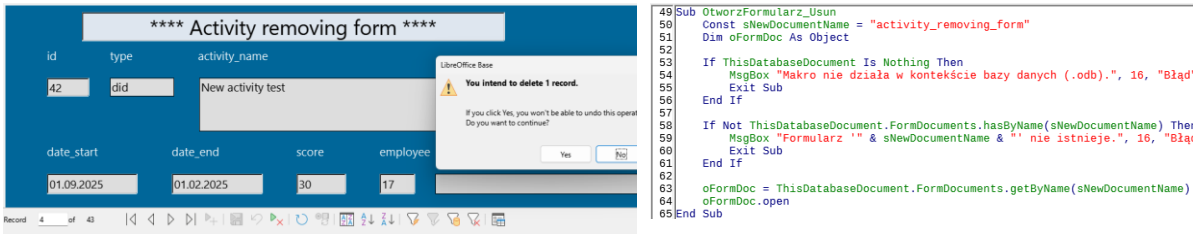


Fig 12. Activity modification form – checking the database constraints

The AEP panel administrator can also identify and remove an activity from the database. To achieve this, they select the appropriate activity in the records panel (Fig. 13a) and press the button. Removing the selected activity is assigned to the *REMOVE form* button (Fig. 8), which invokes the *activity_removing_form* subform. Additionally, before deleting the record from the database, a confirmation window is displayed (Fig. 13a). After confirmation, a macro associated with the form is run, the code of which is shown in Fig. 13b. Verification of the deletion of the selected record from the database was performed using standard SQL queries in *pgAdmin 4*, and the results of this query are shown in Fig. 13c, where the missing record with *ID = 42* can be observed.



a) b)

```
SELECT * FROM public.valid_act
WHERE id_osoby=17
ORDER BY 1 ASC;
```

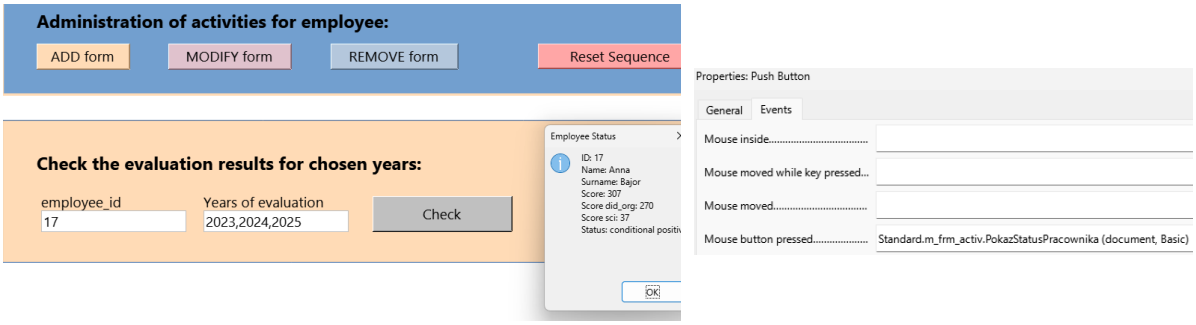
Output Messages Notifications

id [PK] integer	type type_act	activity_name character varying (250)	date_start date	date_end date	score numeric (3)	id_osoby integer
41	sci	example activity: Membership in the recruitment committee: departmental – 202...	2024-01-01	2024-12-30	37	17
43	org	tested: organisation of the conference	2023-01-01	2023-01-01	40	17
44	org	test 2024	2024-01-01	2024-01-02	100	17
45	org	org tested	2025-01-01	2025-12-31	100	17

c)

Fig 13. Activity removing form layer (a), macro (b) and SQL query verifying removed data from database (c)

The second part of the AEP form focusses on the initial analysis of an employee's evaluation based on the criteria outlined in the Rector's Ordinance (2024), and it includes fields that must be completed with the employee's ID and the years to be evaluated. The place during the edition of the form for *CHECK button* action programming is shown in Fig. 14b.



a) b)

Fig 14. Evaluation results: points and actual evaluation status (a) and the action that invokes the macro (b)

Information about the evaluation results for the employee and the evaluation period is presented in Fig. 14. The macro calls the function (Fig. 16) from the database and returns the results in the form of an information window (Fig. 14a). For the sake of speed and optimisation of the EES system operation, this part of the form uses a database object: the *employee_score()* function (Fig. 15), and to verify the operation of the macro (Fig. 16), standard SQL queries containing the above-mentioned function in the code were used.

employee_score(id_pracownika integer, lata integer[])

General Definition Code Options Parameters Security SQL

```
DECLARE
    liczba_lat INTEGER := cardinality(lata);
    prog_calkowity INTEGER := 100 * liczba_lat;
    prog_sci INTEGER := 60 * liczba_lat;
BEGIN
    RETURN QUERY
    SELECT
        p.id_osoby,
        p.name,
        p.sname,
        SUM(a.score) AS suma_punktow,
        SUM(CASE WHEN a.type IN ('did', 'org') THEN a.score ELSE 0 END) AS suma_did_org,
        SUM(CASE WHEN a.type IN ('sci', 'sci_pub', 'sci_conf') THEN a.score ELSE 0 END) AS suma_sci,
        CASE
            WHEN SUM(a.score) >= prog_calkowity AND SUM(CASE WHEN a.type IN ('sci', 'sci_pub', 'sci_conf') THEN a.score ELSE 0 END) >= prog_sci THEN 'positive'
            WHEN SUM(a.score) >= prog_calkowity THEN 'conditional positive'
            ELSE 'negative'
        END AS status
    FROM
        valid_act a
    JOIN
        employee p ON a.id_osoby = p.id_osoby
    WHERE
        (id_pracownika = 0 OR p.id_osoby = id_pracownika)
        AND EXTRACT(YEAR FROM a.date_end) = ANY(lata)
    GROUP BY
        p.id_osoby, p.name, p.sname
    ORDER BY
        p.id_osoby;
END;
```

```
SELECT id_osoby id_employee, imie name, nazwisko sname, suma_punktow total_score, suma_sci sci_score, suma_did_org did_or_score
  FROM employee_score(17,ARRAY[2023,2024,2025])
union
SELECT id_osoby id_employee, imie name, nazwisko sname, suma_punktow total_score, suma_sci sci_score, suma_did_org did_or_score
  FROM employee_score(20,ARRAY[2023,2024,2025])
union
SELECT id_osoby id_employee, imie name, nazwisko sname, suma_punktow total_score, suma_sci sci_score, suma_did_org did_or_score
  FROM employee_score(22,ARRAY[2023,2024,2025]);
```

Output Messages Notifications

SQL Showing rows: 1 to 3 Page

id_employee integer	name character varying	sname character varying	total_score numeric	sci_score numeric	did_or_score numeric
20	Piotr	Poczurkiewicz	90	0	90
17	Anna	Bajor	307	37	270
22	Szymon	Berski	363	318	45

Fig 15. Function for calculating the scores and verification of employee function *employee_score()* executed manually for verifying macro for AEP form

```
Sub PokazStatusPracownika(oEvent As Object)
    Dim oForm As Object
    Dim idPracownika As Integer
    Dim lataTekst As String
    Dim lataTablica() As String
    Dim sql As String
    Dim oContext As Object
    Dim oDB As Object
    Dim oConn As Object
    Dim oStmt As Object
    Dim oResult As Object
    Dim wynik As String

    ' take form from button
    oForm = oEvent.Source.Model.Parent

    ' take value from formula
    idPracownika = oForm.getByName("pole_id_pracownika").Text
    lataTekst = oForm.getByName("pole_lata").Text   ' np. "2022,2023,2024"
    lataTablica = Split(lataTekst, ",")

    ' build SQL query
    sql = "SELECT * FROM employee_score(" & idPracownika & ", ARRAY[" & Join(lataTablica, ",") & "])"

    ' connect to database
    oContext = CreateUnoService("com.sun.star.sdb.DatabaseContext")
    oDB = oContext.getByName(ThisDatabaseDocument.URL)
    oConn = oDB.getConnection("", "")
    oStmt = oConn.createStatement()

    ' execute query
    oResult = oStmt.executeQuery(sql)

    ' create result
    wynik = ""
    Do While oResult.next()
        wynik = wynik & "ID: " & oResult.getString(1) & Chr(13)
        wynik = wynik & "Name: " & oResult.getString(2) & Chr(13)
        wynik = wynik & "Surname: " & oResult.getString(3) & Chr(13)
        wynik = wynik & "Score: " & oResult.getString(4) & Chr(13)
        wynik = wynik & "Score did_org: " & oResult.getString(5) & Chr(13)
        wynik = wynik & "Score sci: " & oResult.getString(6) & Chr(13)
        wynik = wynik & "Status: " & oResult.getString(7) & Chr(13) & Chr(13)
    Loop

    MsgBox wynik, 64, "Employee Status"
End Sub
```

Fig 16. Macro for executing the query in the database and retrieving the data

The macro shown in Fig. 16 contains the following blocks:

- a macro declaration, which consists of a macro identifier that connects the form object, e.g., the *CHECK button*, to the macro (Fig. 16), along with parameters: the *oEvent* argument and the *Object* type,
- variables, e.g., *employee_id*, *years*, *sql_query*, *oDB*, *oConn* etc. along with types,
- defining the definition and location of the form *oForm = oEvent.Source.Model.Parent* along with button controls, e.g., *lataTekst = oForm.getByName("field_years").Text*, which enables retrieving data from the form,
- database operation blocks:
    - constructing an SQL query, e.g., *sql = "SELECT * FROM employee_score(" &…* , which runs a query with a function along with appropriately formatted parameters for this function, e.g., years for employee evaluation in the format: *ARRAY[" & Join(yearsTablica, ",") & "]*,
    - connecting to the database: *oContext =…, oDB =…, oConn = oDB.getConnection("",* "") and *oStmt = oConn.createStatement()*,
    - executing the query: *oResult = oStmt.executeQuery(sql)*,
- retrieving results from the query: *oResult.next() result = result & "ID: " & oResult.getString(1) & Chr(13)* in a *do ..while* loop,
- creating an object displaying information in the form, e.g., *MsgBox result, 64, "Employee Status*".

The execution of this macro initiated by the form control action is shown in Fig. 14a.

Summary of the analysis

Before upgrading the EES system and creating the AEP, a script was created containing the entire database schema, along with all database objects, on which the EES system and the AEP module are based. The *pg_dump* software was used for this purpose. The database schema was then restored from this script to the updated (see Table 1) DBMS. After the restoration, no problems were observed when restoring the database schema to the new server version.

The use of macros allowed for direct execution of SQL database queries (in this case, *INSERT*, *UPDATE*, and *DELETE* queries), as well as for invoking functions directly from a button or other control on the AEP form. This clearly allows for all operations necessary to manage employee activity. The developed model also allows for data validation before saving, blocking invalid values, and entering error messages (Figs. 11-13). Macros allow for programming the button to open other subforms (Fig. 9).

The system also allows for the recording of all changes made to the database by using additional entities in the database's logical schema that collect information about these changes (the *history_t* table in Fig. 4). Each change to the database state by an AEP user, thanks to authorisation (the *userid* attribute in the *history_t* table), is automatically recorded in the *history_t* table by the appropriate *valid_act_history* trigger.

Summary and Conclusions

This article presents a complete model of a university employee evaluation system, created using a combination of a PostgreSQL database and LibreOffice, using macros written in LibreOffice Basic. AEP within the EES System extends the system presented in the works by Berski S. at al (2023) and Berski S. at al (2025). The designed system is an example demonstrating the possibilities of using open source software instead of solutions requiring a license fee. Nowadays, many institutions are seeking to eliminate the use of paid tools, so the search for well-functioning alternative solutions is justified (especially when the institution has the resources of both specialists (IT department) and hardware resources (server infrastructure).

The EES system and the AEP module have been made more flexible and able to handle future changes in the law and the organisation by adapting to new legal and organisational conditions, such as changing the evaluation period and adding new requirements, like making it clear what kind of employment the employee has.

The use of macros allowed us to ensure the full functionality of the designed system, including the execution of CRUD operations on the database and analysis of obtained results, as well as the full utilization of the forms in LO Base. A significant advantage of using macros is the ability to adapt the system's functionality to subsequent changes, which also ensures security in the context of continuous access to the system in compliance with currently applicable regulations. The architecture, based on transferring processing logic from the presentation layer (LibreOffice Base) to the data layer (PostgreSQL) via macros in LibreOffice Basic, reduced the computing resources of the client computers. The implemented solution means that operations with high computational complexity – including queries with joins, recursive CTEs and multi-table transactions – are performed entirely on the server side, taking advantage of its significantly higher computing power and memory.

The designed module also enables the following security measures for the research and teaching unit:

- utilisation of institutional resources (CUT), including IT department and IT infrastructure;
- independence from commercial office software licences;
- reduction of EES system operating costs;
- increased employee safety in terms of work planning and organisation (a priority of the new rector).

References

- API documentation of LibreOffice. [Online], [Retrieved November 15, 2025], https://api.libreoffice.org.
- Berski, S., Wojtyto, D. and Michalik, J. (2023), 'The use of relational data model for employee evaluation: The Case of a University in Poland', Proceedings of the 42nd International Business Information Management Association (IBIMA), ISBN: 978-8-9867719-0-8, 23–23 November 2023, Seville.
- Berski, S., Wojtyto, D., Michalik, J. and Krakowiak, M. (2025), 'Implementation of a Model for Employee Evaluation on the Base of Open-Source Tools: The Case of Libre Office Base', Artificial Intelligence and Machine Learning. IBIMA-AI 2024. Communications in Computer and Information Science, vol 2300, Soliman, K.S. (ed), Cham: Springer Nature Switzerland.
- Golomingi, R., Ebert, L., Ehrlich, C. and Franckenberg, S. (2025), 'Big data in forensic medicine - development of a first forensic pathology database', *Australian Journal of Forensic Sciences*, 1–7.
- Inter-Institutional Committee for Informatics, Conclusions of the discussion on Open Source Software and on the alignment of Open Source Software strategies, Brussels, EU, 2010 [Online], [Retrieved November 15, 2025], https://commission.europa.eu/document/download/c164d210-b52e-4f31-bc1a-e8ee0b817c14_en?filename=declaration_cii-oss.pdf
- Klaasse, J. R., Alewijnse, L. C. and van Der Weerd, J. (2021), 'TraceBase; A database structure for forensic trace analysis', *Science & Justice*, 61(4), 410-418.
- Libre Office Base Guide 7.2. [Online], [Retrieved November 15, 2025], https://books.libreoffice.org/en/BG72/BG72.html
- Libre Office Getting Started Guide 7.0 (2020). [Online], [Retrieved November 15, 2025], https://books.libreoffice.org/en/GS70/GS70.html
- Pitonyak, A. D. (2004), OpenOffice. org macros explained, Hentzenwerke.

- Rector’s Ordinance, Zarządzenie Rektora Politechniki Częstochowskiej nr 339/2022 z dnia 13.12.2022 w sprawie wprowadzenia Kryteriów oraz trybu postępowania podmiotu dokonującego oceny okresowej nauczycieli akademickich za rok 2023 i 2024, (in Polish). [Online], Retrieved November 19, 2025], https://pcz.pl/pracownik/dzial-nauki/ocena-nauczycieli-akademickich.
- Rector’s Ordinance, Zarządzenie Rektora Politechniki Częstochowskiej nr 50/2024 z dnia 23.12.2024 w sprawie wprowadzenia Kryteriów oraz trybu postępowania podmiotu dokonującego oceny okresowej nauczycieli akademickich za rok 2025 w postaci aktualizacji Zarządzenia nr 339/2022 Rektora Politechniki Częstochowskiej z dnia 13 grudnia 2022 roku, polegającego na wydłużeniu okresu oceny o rok 2025, (in Polish). [Online], [Retrieved November 19, 2025],
- https://pcz.pl/pracownik/dzial-nauki/ocena-nauczycieli-akademickich.
- The Rising Cost-Effectiveness of Open-Source Solutions (2024). [Online], [Retrieved November 15, 2025], https://moldstud.com/articles/p-exploring-the-benefits-of-open-source-software
- Tridgell, J. (2025), ‘Open or closing doors? The influence of ‘digital sovereignty’ in the EU’s Cybersecurity Strategy on cybersecurity of open-source software’, *Computer Law & Security Review*, 56, 106078.