CONSTRUCTION OF SPECIALIZED COMPUTER AIDED SOFTWARE ENGINEERING (CASE) TOOLS FOR THE DEVELOPMENT OF EXPERT SYSTEMS

Rustam A. Burnashev
Kazan Federal University, (Russia).
E-mail: r.burnashev@inbox.ru ORCID: https://orcid.org/0000-0001-9548-2943

Albert V. Gubaydullin
Kazan Federal University, (Russia).
E-mail: Global@ores.su ORCID: https://orcid.org/0000-0002-3492-3642

Arslan I. Enikeev
Kazan Federal University, (Russia).
E-mail: Belova-t@ores.su ORCID: https://orcid.org/0000-0002-6033-8498

Recepción: 17/12/2019 **Aceptación:** 03/02/2020 **Publicación:** 13/03/2020

Citación sugerida:
Burnashev, R., Gubaydullin, A., y Enikeev, A. (2020). Construction of specialized computer aided software engineering (CASE) tools for the development of expert systems. *3C Tecnología. Glosas de innovación aplicadas a la pyme, 9*(1), 61-77. http://doi.org/10.17993/3ctecno/2020.v9n1e33.61-77

ABSTRACT

This report presents an approach to building specialized computer-aided software engineering (CASE) tools for the development of expert systems. These tools form an integrated development environment allowing the computer aided development of different applications in the appropriate field. The integrated environment which we consider in our report represents the combination of SWI-PROLOG and Data Base Management System (DBMS) PostgreSQL tools. SWI-PROLOG provides the most appropriate tools for the solution of logical tasks in expert systems. However, SWI-PROLOG cannot manage large amounts of data. Therefore, we need to apply an appropriate data base management system to extend the capability of the knowledge base. For this purpose we used the most advanced open source PostgreSQL tools. As a result of our research we have created tools enabling the compatibility of SWI-PROLOG and DBMS PostgreSQL within the integrated development environment.

KEYWORDS

CASE tools, SWI-PROLOG, PostgreSQL database management system, Expert system.

1. INTRODUCTION

Intelligent information systems and technologies are ones of the most promising and rapidly developing fields in theoretical and applied information technologies. It has had a significant impact on all areas of research and technology related to the use of computers, and it already today gives society what is expected from science - practically meaningful results, many of which contribute to cardinal changes in their applications (Vladimir, 2009). Expert systems (ES) occupied a special place in the development and use of intelligent information systems.

Various types of software tools can be used to create ES, among which SWI-PROLOG seems to be the most appropriate. However, SWI-PROLOG cannot manage large amounts of data. Therefore, we need to include an appropriate data base management system to boost the potential of the knowledge base.

For this purpose we used the most advanced open source PostgreSQL tools. As a result of our research we have created tools enabling the compatibility of SWI-PROLOG and DBMS PostgreSQL within the integrated development environment.

2. METHODS

The report describes the tools and methods which were used to create an integrated development environment.

The integrated development environment includes:

- SWI-PROLOG.
- XPCE for the graphics component.
- PostgreSQL.
- ODBC Driver PostgreSQL DBMS.

SWI-PROLOG is an open release of Prolog. Being formed from the initial data in the form of a chain of reasoning (decision rules) from the knowledge base ES can decide in unique situations for which the algorithm is not known in advance. What is more, problem solution is expected to be carried out in conditions where the initial information is incomplete, unreliable, and ambiguous, during qualitative process assessment (Yuriy, 2004). PROLOG tools appear to be the most appropriate to the solution of the above mentioned problems.

To develop graphics applications, the SWI-PROLOG distribution package includes tools that enable the development of a graphical user interface. These tools for SWI-PROLOG are provided by XPCE.

XPCE is a platform-independent tool for SWI-PROLOG, Lisp and other interactive dynamically typed programming languages.

T is a toolkit for developing graphical applications in Prolog and other interactive and dynamically typed languages. XPCE follows a rather unique approach of for developing GUI applications, which we will try to summarise using the points below.

2.1. ADD OBJECT LAYER TO PROLOG

XPCE's kernel is an object-oriented engine that allows for the definition of methods in multiple languages. The built-in graphics are defined in C for speed as well as to define the platform-independence layer. Applications, as well as some application-oriented libraries are defined as XPCE-classes with their methods defined in Prolog.

Prolog-defined methods can receive arguments in native Prolog data, native Prolog data may be associated with XPCE instance-variables and XPCE errors are (selectively) mapped to Prolog exceptions. These features make XPCE a natural extension to your Prolog program.

2.2. HIGH LEVEL OF ABSTRACTION

XPCE's graphical layer provides a high abstraction level, hiding details on event-handling, redraw-management and layout management from the application programmer, while still providing access to the primitives to deal with exceptional cases.

2.3. EXPLOIT RAPID PROLOG DEVELOPMENT CYCLE

Your XPCE classes are defined in Prolog and the methods run naturally in Prolog. This implies you can easily cross the border between your application and the GUI-code inside the tracer. It also implies you can modify source-code and recompile while your application is running.

2.4. PLATFORM INDEPENDENT PROGRAMS

XPCE/Prolog code is fully platform-independent, making it feasible to develop on your platform of choice and deliver on the platform of choice of your users. As SWI-Prolog saved-states are machine-independent, applications can be delivered as a saved-state. Such states can be executed transparently using the development-environment to facilitate debugging or the runtime emulator for better speed and space-efficiency (Merrit, 1989).

2.5. POSTGRESQL

PostgreSQL, often simply Postgres, is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards compliance. As a database server, its primary functions are to store data securely and return that data in response to requests from other software applications. It can handle workloads ranging from small single-machine applications to large Internet-facing applications (or for data warehousing) with many concurrent users; on macOS Server, PostgreSQL is the default database; (Wierse, Grinstein, & Lang, 1996; Shoham, 2014; Bench-Capon, 1990) and it is also available for Microsoft Windows and Linux (supplied in most distributions).

PostgreSQL is ACID-compliant and transactional. PostgreSQL has updatable views and materialized views, triggers, foreign keys; supports functions and stored procedures, and other expandability (Habarov, 2013).

PostgreSQL is developed by the PostgreSQL Global Development Group, a diverse group of many companies and individual contributors.It is free and open-source, released under the terms of the PostgreSQL License, a permissive software license (Machad, Souza, & Catapan, 2019).

Connection between PostgreSQL and Prolog is provided by an ODBC driver.

An ODBC driver uses the Open Database Connectivity (ODBC) interface by Microsoft that allows applications to access data in database management systems (DBMS) using SQL as a standard for accessing the data. ODBC permits maximum interoperability, which means a single application can access different DBMS. Application end users can then add ODBC database drivers to link the application to their choice of DBMS (Guida & Tasso, 1989).

The ODBC solution for accessing data led to ODBC database drivers, which are dynamic-link libraries on Windows and shared objects on Linux/UNIX. These drivers allow an application to gain access to one or more data sources. ODBC provides a standard interface to allow application developers and vendors of database drivers to exchange data between applications and data sources (Companys, Falster, & Burbidge, 1990; Ardeleanu, 2016).

3. RESULTS

The following basic functions were developed, which enable work with PostgreSQL DBMS inside SWI-PROLOG.

3.1. CONNECTING TO A DATABASE SERVER

```
openDatabase :-

odbc_connect('PostgreSQLProlog',    _,    [    user(postgres),
password('1234'), alias(PostgreSQLProlog), open(once)]).
```

3.2. DISCONNECTING FROM THE DATABASE SERVER

```
closeDatabase:-

odbc_disconnect(PostgreSQLProlog).
```

3.3. GETTING THE LIST OF TABLES FROM THE CONNECTED DATABASE

```
getTables:-

odbc_current_table(PostgreSQLProlog, Table),

write(Table).
```

3.4. GET THE LIST OF COLUMNS FROM THIS TABLE

```
getColumns(Table):-

odbc_table_column(PostgreSQLProlog, Table, Column),

write(Table-Column).
```

3.5. RETRIEVE ALL RECORDS FROM SPECIFIED TABLE

```
getList(Table):-

openDatabase,
concat('SELECT * from "', Table, X),

concat(X, '"', Y),

odbc_query(PostgreSQLProlog, Y , Row, [types([integer, string,
string, string])]),

write(Y),

write(Row),

write_in(Row).
```

3.6. RECEIVING A RECORD FROM CURRENT TABLE WITH A SPECIFIED ID

```
getList(Table, Id):-

openDatabase,

concat('SELECT * from "', Table, X),

concat(X, '"', Y),

odbc_query(PostgreSQLProlog, Y , Row, [types([integer, string,
string, string])]),

write(Y),

write(Row),

write_in(Row).
```

3.7. SELECTION OF TABLE RECORDS FOR A SPECIFIED FILTER

```
selectList(Table, Name, Value):-

openDatabase,

concat('SELECT from "', Table, X1),

concat(X1, '" WHERE ', X2),

concat(X2, Name, X3),

concat(X3, ' = ', X4),

concat(X4, Value, X),

odbc_query(PostgreSQLProlog, X, Row, [types([integer, string,
string, string])]),

write(Row).
```

3.8. DELETE RECORDS FOR THE SPECIFIED FILTER

```
deleteList(Table, Name, Value):-

openDatabase,

concat('DELETE from "', Table, X1),

concat(X1, '" WHERE ', X2),

concat(X2, Name, X3),

concat(X3, ' = ', X4),
concat(X4, Value, X),

odbc_query(PostgreSQLProlog,  X,  Affected,  [types([integer,
string, string, string])]),

write(Affected).
```

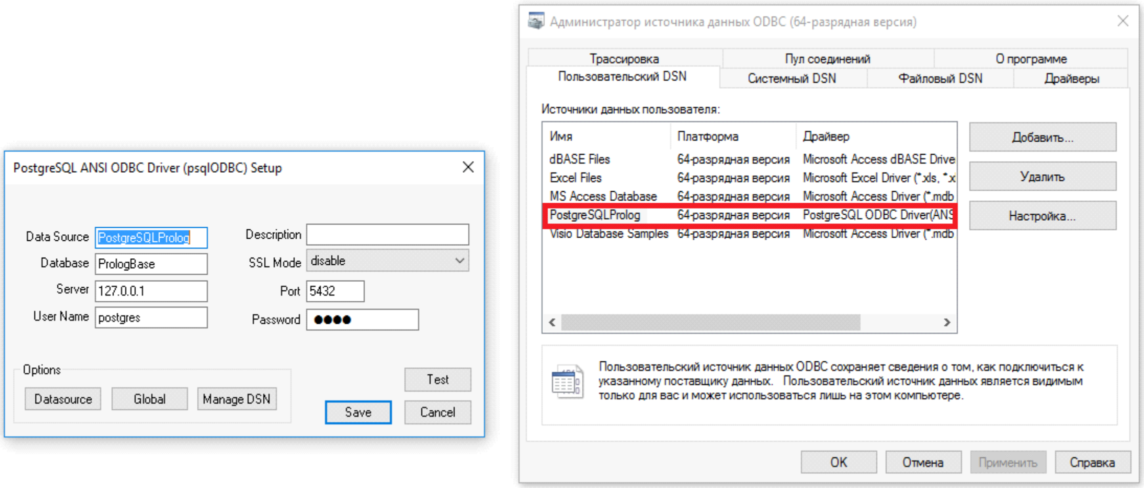
4. DISCUSSION

4.1. PROLOG AND POSTGRESQL INTERACTION

In order to interact and manage XPCE objects from within the SWI-PROLOG kernel environment, the necessary predicates are added to the program, such as:

- new (? Reference, + Class (... Arg ...))
- send (+ Reference, + Method (... Arg ...))
- get (+ Reference, + Method (... Arg ...), -Result)
- free (+ Reference)

ODBC (Open Database Connectivity) is an application interface (API) for providing access to databases (a MICROSOFT product). In order to access to the database, it is necessary to select the data source in the "Data source administrator" window.


PostgreSQL ANSI ODBC Driver (psqlODBC) Setup

Data Source	PostgreSQLProlog	Description	
Database	PrologBase	SSL Mode	disable
Server	127.0.0.1	Port	5432
User Name	postgres	Password	●●●●

Администратор источника данных ODBC (64-разрядная версия)

Источники данных пользователя:

Имя	Платформа	Драйвер
dBASE Files	64-разрядная версия	Microsoft Access dBASE Drive
Excel Files	64-разрядная версия	Microsoft Excel Driver (*.xls, *.x
MS Access Database	64-разрядная версия	Microsoft Access Driver (*.mdb
PostgreSQLProlog	64-разрядная версия	PostgreSQL ODBC Driver(ANS
Visio Database Samples	64-разрядная версия	Microsoft Access Driver (*.mdb

Пользовательский источник данных ODBC сохраняет сведения о том, как подключиться к указанному поставщику данных. Пользовательский источник данных является видимым только для вас и может использоваться лишь на этом компьютере.

Figure 1. ODBC Data Source Administrator.

As the result of the research we have developed a knowledge base and tools for the selection of data from databases using the logical programming language SWI-PROLOG and DBMS PostgreSQL.

Based on the necessary predicates enabling the compatibility of SWI-PROLOG and PostgreSQL, the following files were created, namely:

- UserInterface.pl - User interface.
- LocalBase.pl - Local Database.
- DataConnection.pl - Contains the functions for interaction with the ODBC driver.

In the future, it is also necessary to envisage the possibility of adding not only a certain set of data to the knowledge base, but also adding new inference rules not existing at the time of system development. Thus, all this confirms the fact that the development of a fully-fledged expert system of this kind is a complex and expensive task.

4.2. SQL DATA REPRESENTATION IN PROLOG

Databases have a poorly standardized, but very rich set of data types. Some types of data have analogues with PROLOG. To fully correlate with data types when developing CASE tools, you need to define PROLOG data types for SQL types that do not have a standard analog with PROLOG (for example, timestamp). For example, many variants of the SQL DECIMAL type cannot be mapped to an integer number of PROLOG. However, matching to an integer can be the right choice for an application. That's why it is important to understand how SQL data types can be represented in Prolog.

The PROLOG/ ODBC interface defines the following types of PROLOG data with the specified default conversion.

4.3. ATOM

It is used by default for SQL types char, varchar, longvarchar, binary, varbinary, longvarbinary, decimal and numeric. Can be used for all non-structural types.

4.4. STRING

A string of the extended SWI-PROLOG type.

4.5. CODES

List of code characters. Can consist of any amount of text.

4.6. INTEGER

Used by default for the SQL, tiny int, small int and integer bit types.

4.7. FLOAT

Used by default for SQL real, float and double types.

4.8. DATE

The PROLOG date (Year, Month, Day) of form used by default for SQL dates (Year, Month, Day).

4.9. TIME

A PROLOG term for the time format (Hour, Minute, Second) used by default by SQL.

4.10. TIMESTAMP

In the PROLOG language, the term 'timestamp' (Year, Month, Day, Hour, Minute, Second, Fraction) is used by default for SQL type timestamps.

5. SUMMARY

On the basis on conducted research the following program and GUI was developed:

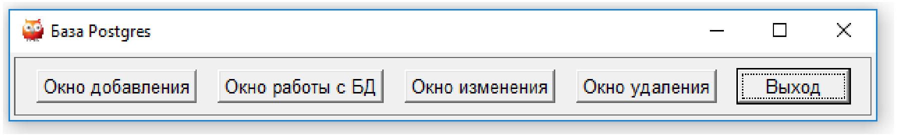


Figure 2. The main window of the program

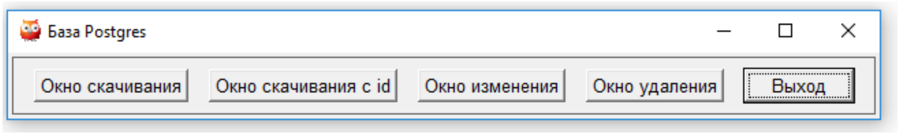


Figure 3. The menu for working with database

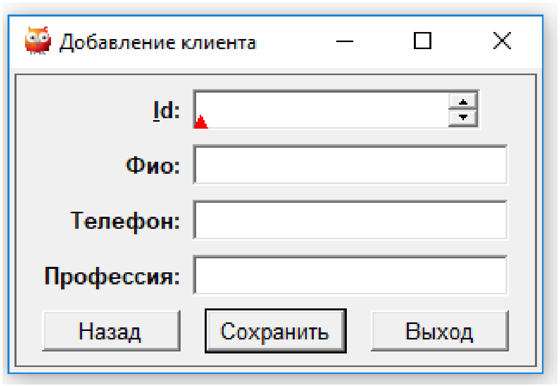


Figure 4. Adding form

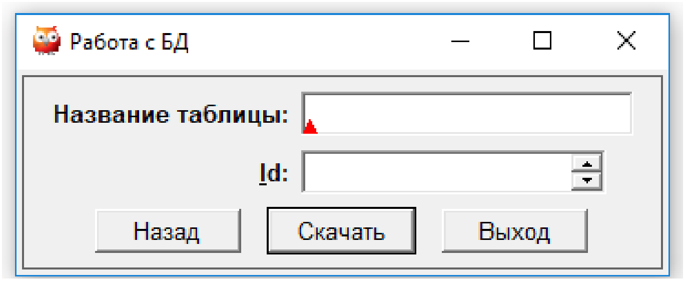


Figure 5. Downloading from database form

Program, which is presented in this report, is an Open Source management tool for Postgres and Prolog. It can download, edit and delete data from PostgreSQL database. User can manually choose the database to work with. After the download, all information is stored in .pl file and ready to work. Using this program user also can add, search, edit and delete rows from this file. So we think, That this program can probably become a part or a module of complete CASE tool software to work with both PostgreSQL and Prolog. It is designed to meet the needs of both novice and experienced Postgres users alike, providing a powerful graphical interface that simplifies the creation, maintenance and use of PostgreSQL databases.

6. CONCLUSIONS

An important feature of the expert system is that the user cannot only receive a consultation, but also to access all the knowledge from system storage by asking relevant questions.

The use of expert systems makes it possible to make decisions in unique situations for which the algorithm is not known in advance and is formed from the initial data in the form of a chain of reasoning (decision rules) from the knowledge base.

The novelty of the new CASE tool presented is that it ensures the compatibility of SWI-PROLOG and DBMS PostgreSQL within the framework of a single integrated development environment. In the future this will be applicable to the development of various expert systems.

ACKNOWLEDGEMENTS

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

REFERENCES

Ardeleanu, S. (2016). *Relational Database Programming: A Set-Oriented Approach.* Apress, p. 149.

Bench-Capon, T. J. (1990). *Knowledge representation: An approach to artificial intelligence* (Vol. 32). Elsevier.

Companys, R., Falster, P., & Burbidge, J. L. (Eds.). (1990). *Databases for production management*. Elsevier.

Guida, G., & Tasso, C. (1989). *Topics in Expert System Design: Methodologies and Tools (Studies in Computer Science and Artificial Intelligence)*. Elsevier Science Inc.

Habarov, P. (2013). *PROLOG – The Language Of Intelligent And Expert Systems Development: A Study Guide*. SPGLTU.

Machado, A. D. B., Souza, M. J., & Catapan, A. H. (2019). Systematic Review: Intersection between Communication and Knowledge. *Journal of Information Systems Engineering and Management, 4*(1), em0086. https://doi.org/10.29333/jisem/5741

Merrit, D. (1989). *Building Expert Systems in Prolog.* Springer-Verlag New York Inc.

Shoham, Y. (2014). *Artificial intelligence techniques in Prolog*. Morgan Kaufmann Publishers.

Vladimir, S. (2009). *Toiskin: Intelligent Information Systems: A Study Guide*. SGPI, Stavropol.

Wierse, A., Grinstein, G. G., & Lang, U. (Eds.) (1996). *Database Issues for Data Visualization.* Springer-Verlag Berlin Heidelberg, p. 263.

Yuriy, F. (2004). *Telnov: Intelligent Information Systems*. Moscow International Institute of Econometrics, Computer Science, Finance and Right.