# Specialized Case Tools for the Development of Expert Systems

Rustam A. Burnashev(✉), Albert V. Gubajdullin, and Arslan I. Enikeev

Kazan Federal University, Kazan, Tatarstan, Russia
r. burnashev@inbox. ru

**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 and 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 [1]. 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.

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


Á. Rocha et al. (Eds.): WorldCIST'18 2018, AISC 745, pp. 599–605, 2018.
https://doi.org/10.1007/978-3-319-77703-0_59

The integrated development environment includes:

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

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 make a decisions 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 [3]. 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.

## 2 Methods

XPCE is a platform-independent tool for SWI-PROLOG, Lisp and other interactive dynamically typed programming languages. This framework received the greatest popularity in the Prolog language.

In order to interact and manage XPCE objects from within the SWI-PROLOG kernel environment, the necessary predicates are added to the program [2], 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 (Fig. 1).

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.

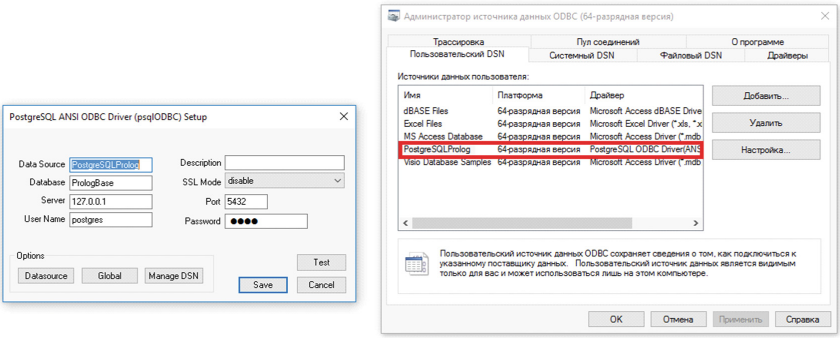


**Fig. 1.** ODBC data source administrator.

## 3 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 a particular application.

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

**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.

**string**
A string of the extended SWI-PROLOG type.

**codes**
List of code characters.

**integer**
Used by default for the SQL, tinyint, smallint and integer bit types.

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

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

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

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

## 4 Results and Discussions

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

**Connecting to a Database Server**

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

**Disconnecting from the database server**

```
closeDatabase:-
odbc_disconnect(PostgreSQLProlog).
```

**Getting the list of tables from the connected database**

```
getTables:-
odbc_current_table(PostgreSQLProlog,Table),
write(Table).
```

**Get the list of columns from this table**

```
getColumns(Table):-
odbc_table_column(PostgreSQLProlog,Table,Column),
write(Table-Column).
```

**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).
```

**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).
```

**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).
```

**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, Affect-
ed,[types([integer,string,string,string])]),
write(Affected).
```

On the basis on conducted research the following GUI was developed (Figs. 2, 3, 4 and 5):

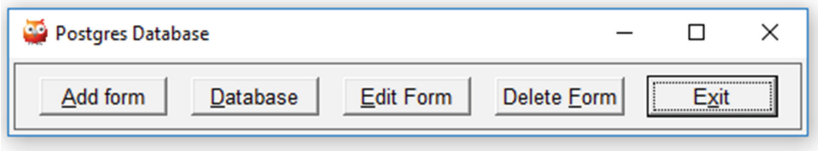


**Fig. 2.** The main window of the program

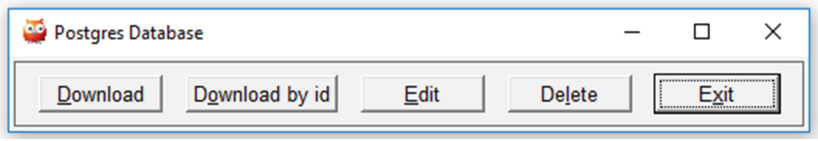


**Fig. 3.** The menu for working with database

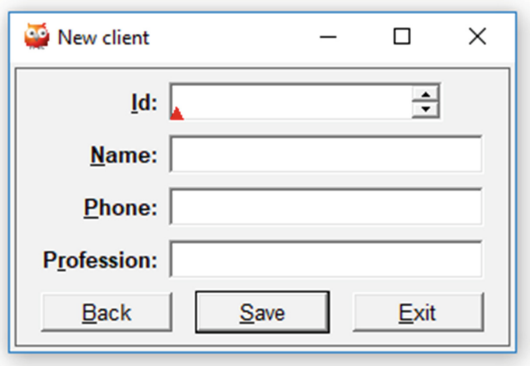


**Fig. 4.** Adding form

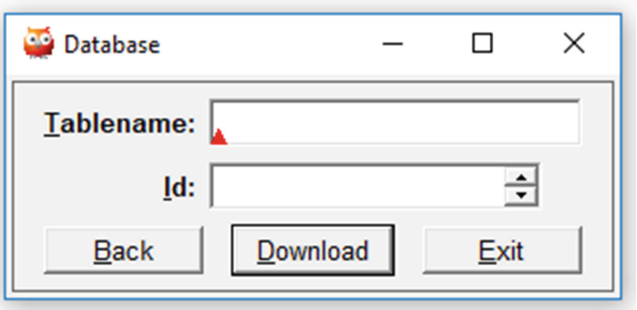


**Fig. 5.** Downloading from database form

## 5 Conclusion

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.

## References

1. Toiskin, V.S.: Intelligent Information Systems: A Study Guide. SGPI, Stavropol (2009)
2. Habarov, P.: PROLOG – The Language of Intelligent and Expert Systems Development: A Study Guide. SPGLTU, Saint-Petersburg (2013)
3. Telnov, Y.F.: Intelligent Information Systems. Moscow International Institute of Econometrics, Computer Science, Finance and Right, Moscow (2004)
4. Muller, H.A., Norman, R.J., Slonim, J.: Computer Aided Software Engineering. Springer, New York (1996)