Relational Data Bases

' a la carte'

Miguel Filsueiras
Luis Moniz Pereira

Nucleo de Inteligencia Artificial
Departamento de Informatica
Universidade Nova de Lisboa
Quinta da Torre
2825 Monte de Caparica
PORTUGAL

Abstract

We have developed a general purpose program (written in Prolog) which uses information gathered interactively from the user to generate specific menu-based consultation programs, tailored to suit the relational data base and access requirements of each application. Every menu allows for quite general relational queries, comprising universal and existential quantifications, conjunctions, disjunctions, and negation as non-provability. Some of the relational data base access concepts employed concern implicit fields, special access predicates, references to text strings stored on disk, finding complete descriptions from partial ones, etc.

We claim the great usefulness of this program: for those who have data to store and retrieve the only work is to plan a relational data base; the consultation program is almost instantly made.

The use of Prolog was paramount for the ease of design and implementation of this system.
Introduction

We have developed a general purpose program (written in Prolog) which uses information gathered interactively from the user to generate specific menu-based consultation programs, tailored to suit the relational data base and access requirements of each application. Every menu allows for quite general relational queries, comprising universal and existential quantifications, conjunctions, disjunctions, and negation as non-provability. Some of the relational data base access concepts employed concern implicit fields, special access predicates, references to text strings stored on disk, finding complete descriptions from partial ones, etc.

We claim the great usefulness of this program: for those who have data to store and retrieve the only work is to plan a relational data base; the consultation program is almost instantly made.

The use of Prolog was paramount for the ease of design and implementation of this system. Indeed, Prolog as language [W. Cluckin, C. Mellish 81] comprises in itself relational queries, and relational data bases are inherent to it. Additionally, it incorporates a search strategy for data retrieval, besides being a powerful symbolic manipulating language on its own. Thus it is ideally suited for piecemeal program generation; this is so because Prolog clauses are extremely modular and need not have any side-effects. Consequently, the application dependent clauses of the consultation program are simply added to its application independent core.

Furthermore, queries are easily built along successive menu steps because Prolog clauses do not have to return complete data structures, but may cooperate instead to their successive refinement.

The user relational data base is just a Prolog subprogram, in the form of unit clauses, which is added to the consultation part. Updating is simply performed with an editor, which in some systems may be called from within Prolog. The criticism that the address space puts a limit on the data base size is waved in the 32-bit address machines. In smaller address space machines, one can set up several Prolog jobs to hold data base parts, and have them communicate through a message queue handler. We have done so on a PDP 11/23 running under RT11-XM.
Basic Notions

Data are supplied, as in all relational data bases, as n-tuples of arguments of relations (or predicates), and stored as Fraglo unit clauses. Each argument (or field) has a meaning dependent on the predicate and argument position.

For print out purposes, a heading for each field is requested from the user. The consultation program assumes identical headings correspond to similar fields, for all data base relations.

A distinction is drawn between 'output-only' and 'both-way' arguments: the former are only used for output and cannot be instantiated in queries, the latter can be used for data retrieval as well, and so can be instantiated in queries.

We have developed some optional features to increase data base compactness and improve access. They are:

- References to texts. Sometimes there are non-formatted informations that are best kept as texts. We use fields of the form t(File, Number) to refer to a text under the given Number in the given File. Such fields are viewed as output-only and we have a special predicate (in Prolog) to retrieve such texts.

- Implicit fields. It is useful for derived information (implicit fields) to be built from actual fields of a data base predicate, so as to avoid duplication. To this end we provide two different implementations. They should be chosen according to how often the derived information is required. For the implicit fields frequently used, an interface predicate is created that calls the corresponding data base one and builds all such fields. For infrequently used implicit fields, ancillary conditions are employed to define them. Each such condition is only activated whenever the correspondent implicit field is required. A data base predicate may be augmented with both kinds of implicit fields.

- Special access predicates. Information may need some preprocessing before being output or retrieved from an explicit or implicit field. We tackle this by allowing special user defined predicates to be called when accessing such fields. As an example consider the case of lists. One may be interested in obtaining not a list but one of its elements; pretty-printing may also be required.
To build a query one needs to know the type of operation envisaged, the logical connectives and predicates involved, as well as a specification of which fields have an input value imposed and which are to be output. Additionally, a request can be made for similar fields to hold the same value, or for a set of answers to be partitioned relative to the different values of one or more fields.

The first menu presented to the user offers a choice of quantifier/operations as well as calls to subprograms independent from the consultation program proper (e.g. the one that finds complete designations from partial ones - the 'oracle'). Examples of quantifiers are

one all how many

and of operations on numeric fields are

sum mean least value greatest value

Next, for describing the information in question, the user is presented, in a first stage, with two more menus. One allows the choice of a data base predicate (or interface predicate if any), and of a combination of its fields and their mode of access. Another menu follows, to select between launching the query to obtain an answer (the specification is assumed complete) or to connect the partial specification to what follows with an and / andnot / or / ornot operator. The completion of the specification is then resumed from the first stage.

The specification of any field may be a combination of four modes:

- a value is input
- a value is to be output
- output is to be grouped according to its different values
- the information in this field must match the information in similar field(s) (i.e., those with identical output headings) occurring in predicates already incorporated in the query.

All non-contradictory combinations of these can be made, the consultation program rejecting any inconsistency; of course one cannot give two different values to be matched
simultaneously, but can give a value and ask it to be output as well. Output-only fields, too, cannot be input a value.

The following section thoroughly exemplifies these features.

A Consultation Program

We now describe in some detail a real data base system made under a contract with JNICT – 'Junta Nacional de Investigação Científica e Tecnológica' (National Science and Technology Research Council) – regarding data on FACC – 'Fundão de Apoio à Comunidade Científica' (Scientific Community Support Fund) – concerning research centers (about 200), their organisms (one per center and per year), and applications for funding (about 500 per year) [L. Moniz Pereira, M. Filgueiras 82].

The extensional data base predicates were designed as follows:

center(Number_c, Initials, Sector, District, Info_c)

organic(Number_c, Year, Director_Title, Director_Name, Info_o)

application(Number_c, Type, Year, Item, Researchers, Value_applied, Value_granted, Process_no, Status)

where Number_c is the center number, and Info_c, Info_o and Item are references to texts on disk, containing information about the center, the organic and the item(s) referred in the application, respectively. Three special features were used:

- center name: this was made an implicit field of the data base predicate center, and was defined as a reference to text with the form t(cent, Number_c). We used an interface predicate to implement it though the use of an ancillary condition would be more efficient as the field is an output-only one and not so often used

- director name: for output we built another implicit field of the form Director_title : Director_name (where ':' is an infix operator) implemented through an ancillary condition

- researchers: this field is either 0 (zero: no researchers involved or unknown), or of the form
N1+N2+... ( '+' being another infix operator) where N1, N2, ... are researcher numbers. When retrieving the applications a given researcher is involved in, we want to find fields containing his number. When output is wanted from this field, we do not want numbers but names, so we build a list of references to texts that have the form t(researcher, N). Accordingly, we use two special predicates to access the field, one for when it is input, the other for when it must be output.

We illustrate the consultation program with a small fictitious data base. In the protocol below, user answers follow the prompt `!` and commentary comes between braces.

{ the first menu is }

one) all) how)many sum mean greatest least oracle ! bye
    : one
        { give me one }
    a)pplication c)enter organic ! error
    : a
        { application }

n)umber c) type y)ear i)tem r)esearchers
    va)plied vs)anted p)rocess_no s)tatus all)fields ! error
    : n 61
        { no. of center 61
    : y 1980
        in the year 1980,
    : va?
        the value applied,
    : vs?
        the value granted,
    : !
        and nothing else }

a)nswer and andnot or ornot error
    : a
        { answer }

value granting 25
value applied 25

{ return to the initial situation after the answer }

one) all) how)many sum mean greatest least oracle ! bye
    : o

{ the oracle obtains a complete designation of one or more researchers or centers as desired, whose partial designation is known }
r)esearchers c)enters !

write in one line only the partial designation you know
and capitalize proper and common names;
the usual abbreviations are allowed if ended with a dot.

C. de Informatics

identification no. 61
Centro de Informatica Universidade Nova de Lisboa

more another ! { here one may request complete
: m
designations for the same partial one,
or supply another partial designation,
or (!) terminate }

unknown

r)esearchers c)enters ! { consultation ends and the system
: ! returns to the initial situation }

{ give me all about an application from center no. 61 }

one) all) how)many s)um m)ean s)reatest l)east o)race ! bye
: one

a)pplication c)enter o)rganic ! error
: a

number c) type y)ear i)tem r)esearchers

va)plied vs)ANTED P)rocess_no s)tatus all)fields ! error
: n 61
: all
: !

a)nswer and andn)ot or orn)ot error
: a

number c) 61
type 2
year 1980
item visit of David Warren
researchers Luis Moniz Pereira
value applied 25
value granted 25
process_no 347
status ok
{ how many are the applications? }

one all how many sum mean greatest least oracle ! bye
: how
a application c enter organic ! error
  : a
   { a ! would return the system to the initial situation }
number_c type year item researchers
va applied vs ranted p process_no status all fields ! error
  { no further specification is intended ;
  ! is given since the specification has ended }
answer and and not or or not error
: a

number 21
one all how many sum mean greatest least oracle ! bye
: all { give me for all }
answer c
  { centers }
number_c initials sector district in fo c name
all fields ! error
: n?
    { their number and
: i?
    their initials,
: d*
    grouping them by district (*) }
: !
answer and and not or or not error
: a
initials-number_c by district
* lisboa :
cird 76 is 206 spm 7
spb 19 spcv 20 ...
* porto :
spo 40 cemup 15 demfeup 11
deafeup 44
{ give me, for all centers, their name, 
grouping them by district and by sector }

one) all) how many sum mean greatest least oracle ! error
*: all

application center organic ! error
*: c

number initial sector district info name
*: na?
*: s*
*: d*
*: !

answer and and not or or not error
*: a

name by sector-district
* lisboa ipsfl :

Centro de Informatica e Pesquisa Para o Desenvolvimento
Instituto de Soldadura
Sociedade Portuguesa de Matematica

* porto ipsfl :

Sociedade Portuguesa de Ornitologia

* lisboa government :

Direccao-Geral do Saneamento Basico

* lisboa ensino_inic :

Centro de Fisica Nuclear da Universidade de Lisboa
Centro de Informatica Universidade Nova de Lisboa

* porto ensino_inic :

Centro de Engenharia Mecanica da Universidade do Porto

* lisboa ensino :

Departamento de Estudos Classicos

* porto ensino :

Departamento de Engenharia Mecanica
    da Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Quimica
    da Faculdade de Engenharia da Universidade do Porto
a) nswer and a ndn)ot or orn)ot error
   : a

number_c - initials - sector - district - info_c - name

40 spo ipsfl porto address1
  Sociedade Portuguesa de Ornitolologia

15 cemup ensino_inic porto address2
  Centro de Engenharia Mecanica da Universidade do Porto

11 demfeur ensino porto address3
  Departamento de Engenharia Mecanica
d  da Faculdade de Engenharia da Universidade do Porto

44 dequeur ensino porto address3
  Departamento de Engenharia Quimica
d  da Faculdade de Engenharia da Universidade do Porto

{ give me all about each organic in 1980 }

one> all> how>many s>um m>e an d>reatest l>east o>race ! bye
  : all

a) p plication c>enter o>rganic ! error
  : o

number_c y>ear dt>itle dn>ame i>nfo_o d>irector
  all>fields ! error
  : all
  : y 1980
  : !

a) nswer and a ndn)ot or orn)ot error
   : a

number_c - year - info_o - director

11 1980 info1 prof: Vasco Sa
61 1980 info2 prof: Candido Marciano da Silva

one> all> how>many s>um m>e an d>reatest l>east o>race ! bye
  : how

a) p plication c>enter o>rganic ! error
  : error
  { 'how' was not intended ; the program is

[EXECUTION ABORTED] automatically restarted after an error }
Generating Consultation Programs

The consultation program described in the last section was designed from scratch, implemented and thoroughly tested in a man-month on a PDP-11/23 with 128KB central memory and 2 floppy-disks (RX02). Soon it was felt that a 'meta-consultation program' was within reach and would be extremely useful. In fact the data base dependent sections of the consultation program were easily set apart, and provision was made to generate these sections from answers cajoled from the user about his data base. In some 6 man-days the generating program in Prolog was finished and tested.

It is our strong conviction that all this was only possible through the use of Prolog.

Generated programs are concise (contain exactly what is needed to implement the features selected by the user) and to
some extent are protected from errors (e.g., duplicate names). Obviously, the user must provide any Prolog subprogram or special access predicate alluded to when generating the consultation program.

Below, we present a sample session with the generating program, regarding the consultation program of the previous section. Again we use '!' to prompt the user.

Hello!
In case you have any doubt type '?' for help.

output file ? : ?
Please give the name of the file where the consultation program is going to be written to.
output file ? : facc

data base file ? : nucl

password - "no" if none ? : $$$

do you need integers greater than 16383 ? : yeah
acceptable answers - [yes, no]
do you need integers greater than 16383 ? : yes

Now, some questions concerning quantifiers and subprograms called from the 1st menu of the access program.
do you need arithmetic ? : yes
Which of the following do you need -

sum ? : yes
mean ? : yes
greatest value ? : yes
least value ? : yes
do you need other functions ? : no

are there references to texts in the data base ? : yes
do you want to include an 'oracle' ? : ?
The oracle is a subprogram that finds complete designations from partial ones. The complete designations should be grouped into different files according to their meanings - for instance, names of people, organizations,
do you want to include an 'oracle' ? : yes

mnemonic for group of designations ? : c
rest of name ? : enters
file containing this group ? : cent

more groups ? : yes
mnemonic for group of designations ? : r
rest of name ? : researchers
file containing this group ? : researcher

more groups ? : no

do you want to make calls to other subprograms ? : no

Questions concerning database predicates, their fields and access to them -

mnemonic for database predicate ? : c
rest of name ? : enter

predicate name ? : dbase
predicate name already in use ; try another
predicate name ? : center

no. of explicit fields for this predicate ? : 5

do you want an interface predicate for this db one ? : yes

name of the interface predicate ? : cent

no. of implicit fields created by this interface ? : 1

an implicit field is V6
what is the Prolog condition for it - do not use blanks ! - ?
: V6=t(center,V1)
V6=t(center,V1) ok ? : yes

mnemonic for predicate field ? : n
rest of name ? : number_c
heading for output ? : number_c
a normal field ? : ?
A normal field is a both-way field that needs no special predicates to be accessed.
a normal field ? : yes

mnemonic for predicate field ? : n
field mnemonic already in use ; try another

mnemonic for predicate field ? : i
rest of name ? : initials
heading for output ? : initials
a normal field ? : yes

mnemonic for predicate field ? : s
rest of name ? : sector
heading for output ? : sector
a normal field ? : yes

mnemonic for predicate field ? : d
rest of name ? : strict
heading for output ?: district
a normal field ?: no
a both-way field ?: no
an output only field ?: yes
access by special predicate ?: no

mnemonic for predicate field ?: in
rest of name ?: fo_c
heading for output ?: info_c
a normal field ?: yes

mnemonic for predicate field ?: na
rest of name ?: me
heading for output ?: name_c
a normal field ?: no
a both-way field ?: no
an output only field ?: yes
access by special predicate ?: no

any implicit field created by ancillary conditions ?: no

more data base predicates ?: yes

mnemonic for data base predicate ?: o
rest of name ?: rdanic
predicate name ?: organic

no. of explicit fields for this predicate ?: 5

do you want an interface predicate for this db one ?: no

mnemonic for predicate field ?: n
rest of name ?: number_c
heading for output ?: number_c
a normal field ?: yes

mnemonic for predicate field ?: y
rest of name ?: ear
heading for output ?: year
a normal field ?: yes

mnemonic for predicate field ?: dt
rest of name ?: itle
heading for output ?: *
a normal field ?: no
a both-way field ?: no
an output only field ?: no
access by special predicate ?: no

mnemonic for predicate field ?: dn
rest of name ?: ame
heading for output ?: *
a normal field ?: no
a both-way field? : no
an output only field? : no
access by special predicate? : no

mnemonic for predicate field? : i
rest of name? : nfo_o
headings for output? : info_o
a normal field? : no
a both-way field? : no
output only field? : yes
access by special predicate? : no

any implicit field created by ancillary conditions? : yes

mnemonic for field created by an ancillary condition? : d
rest of name? : director
headings for output? : director

result is V6
what is the ancillary condition - do not use blanks! - ?
V6=V3:V4
V6=V3:V4 ok? : yes

more ancillary conditions? : no

more data base predicates? : yes

mnemonic for data base predicate? : a
rest of name? : application
predicate name? : application

no. of explicit fields for this predicate? : 9

do you want an interface predicate for this db one? : no

mnemonic for predicate field? : n
rest of name? : number_c
headings for output? : number_c
a normal field? : yes

mnemonic for predicate field? : t
rest of name? : type
a normal field? : yes

mnemonic for predicate field? : i
rest of name? : item
headings for output? : item
a normal field? : no
a both-way field? : no
an output only field? : yes
access by special predicate? : no
mnemonic for predicate field ? : r
rest of name ? : esearchers
heading for output ? : researchers
a normal field ? : no
a both-way field ? : yes
access by special predicate ? : yes
when a value is input ? : yes

if input value is V and field value is V4
what is the Prolog condition - do not use blanks ! - ?
: res_in(V,V4)
res_in(V,V4) ok ? : yes

and when a value is output ? : yes

if output value is X4 and field value is V4
what is the Prolog condition - do not use blanks ! - ?
: res_names(V4,X4)
res_names(V4,X4) ok ? : yes

mnemonic for predicate field ? : va
rest of name ? : applied
heading for output ? : value_applied
a normal field ? : yes

mnemonic for predicate field ? : vs
rest of name ? : granted
heading for output ? : value_granted
a normal field ? : yes

mnemonic for predicate field ? : p
rest of name ? : process_no
heading for output ? : process_no
a normal field ? : yes

mnemonic for predicate field ? : s
rest of name ? : status
heading for output ? : status
a normal field ? : yes

any implicit field created by ancillary conditions ? : no

more database predicates ? : no

Your consultation program is in file facc
Don't forget appending to it the definitions of the special access predicates you mentioned here.

Bye !
Conclusions

---

Prolog is an excellent unrivaled language for this type of application, but there is still room for improvement through research. In particular, large databases require more indexing facilities, and multi-user access with on-line updating poses special protection problems. Imposing integrity constraints is also beginning to be explored in the context of logic programming. Query planning and more intelligent access mechanisms in general are also in order.

References

---

L.M. Pereira, M. Filgueiras

82 Manual de Utilizacao da Base de Dados FACC
Departamento de Informatica
Universidade Nova de Lisboa

W. Clocksin, C. Mellish

81 Programming in Prolog
Springer Verlag