Лабораторная работа №3

Развитые типы данных в программах на Прологе. Базы данных в программах на Прологе.

ЦЕЛЬ РАБОТЫ:

Целью работы является элементарное введение в базы данных в системе VISUAL PROLOG(VIP). Студент должен ознакомиться и разобраться с операциями создания, открытия, поиска и изменения информации в базе данных и самостоятельно написать учебную программу для закрепления навыков программирования.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.

Данные хранятся в базе данных в форме записей. Запись состоит из полей , представляющих различные типы данных. В Прологе необходимо объявить структурированный тип для записей базы данных - функтор. Пусть в нашей учебной базе содержатся сведения о футбольных командах; полями таких записей будут название команды, страна и международный рейтинг. Объявим тип записей так

```
DOMAINS
name=symbol
country=symbol
rate=integer
teams=teams(name, country, rate)
```

Раздел DOMAINS в Прологе используется для объявления нестандартных или пользовательских типов. Нами объявлены следующие типы: name, country, rate, teams. Наиболее интересен последний тип, который называется функтором. Примером индивидуальной записи такого типа может служить следующая

teams(милан, италия, 75).

Теперь начнем рассматривать команды для работы с базами данных. В Прологе есть два типа баз данных: внешние и внутренние. Последние целиком грузятся в оперативную память, а потому имеют скорее учебное назначение. Поэтому рассмотрим внешние базы данных. Все команды для работы с внешними базами данных делятся на четыре группы.

Первая группа: общие команды для работы с базами данных. Сюда относятся команды создания, открытияб закрытия, удаления, сжатия, копирования и др.

Вторая группа: команды для работы с цепочками записей. Под цепочкой (chain) понимается последовательность записей, имеющих один и тот же тип. Например, выше мы ввели тип teams для записей базы данных, но могли ввести и другие типы, так что получили бы несколько цепочек соответственно.

Третья группа: команды для работы с индивидуальными записями.

Четвертая группа: команды для работы с B-деревьями. B-деревья предназначены для быстрого поиска данных по ключам. В качестве ключей могут использоваться, например, названия команд

Итак, начнем с первой группы.

1) Создание базы данных выполняется командой **db_create**(селектор, имя_файла, место) Селектор является объектом типа **db_selector** и объявляется в разделе **GLOBAL DOMAINS** .Этот раздел должен быть единственным в приложении. Однако система

сама его создает в файле с расширением **PRE**, который она подключает к программе с помощью команды **#include.** Вам придется открыть этот PRE-файл и сделать добавочную запись, как будет объяснено ниже. *Селектор* используется в качестве программного имени базы данных. *Имя файла* – это имя файла на диске. *Место* – это один из следующих стандартных спецификаторов:
in_file - база данных создается в файле и сохраняется в файле;
in_memory – база данных создается в памяти ЭВМ;
in_ems – для создания базы данных используется расширенная память. Мы будем использовать вариант **in_file**.

2) Открытие базы данных выполняется командой
db_open(селектор, имя_файла, место).
3) Закрытие базы данных выполняется командой
db_close(селектор)
4) Удаление базы данных выполняется командой **db_delete**(имя_файла, место).

Заметим, что удалять можно только закрытую базу. Команда создания базы данных одновременно делает ее открытой.

Пример.

```
DOMAINS

        db_selector = mydba

  GOAL
        db_create(mydba, "db.bin",in_file),
    % .....   какие-то действия
        db_close(mydba),
        db_delete("dd.bin", in_file).
```

5) Для открытия поврежденной базы данных используется команда
db_openinvalid(селектор, имя_файла, место).

Повреждение базы может возникнуть при сбое питания и открытой базе, некорректном завершении работы.

6) Для проверки факта обновления базы данных применяется команда
db_update(селектор)
7) Следующая команда выполняет безопасное сохранение базы данных на диске; при этом база данных регистрируется как valid:
db_flush(селектор).

8) Последние команды этой группы:
db_statistics(селектор, число_записей, размер_оперативной_памяти,
размер_БД_в_байтах, свободная_память_на_диске)

Эта команда выдает важную информацию. Здесь
число_записей имеет тип ULONG и получает значение, равное числу записей в базе;
размер_оперативной_памяти имеет тип ULONG и определяет число записей, загруженных в текущий момент в оперативную память;
размер_БД_в_байтах имеет тип ULONG;
свободная_память_на_диске имеет тип ULONG и определяет доступную (незанятую) память на внешнем диске.

Наконец, весьма полезна команда

db_garbagecollect(селектор) , которая собирает в кучу все пустые фрагменты в базе данных, те. предупреждает фрагментацию пространства памяти на диске на множество мелких незанятых участков.

Переходим теперь ко второй группе команд.

Отметим, что в одной и той же базе может быть несколько различных цепочек.

9) Команда выбора следующей по порядку цепочки:
db_chains(селектор, Chain)

Эта команда присваивает переменной Chain имя очередной цепочки. Если цепочек больше нет, то возникает возврат в ближайшую точку ветвления.

10) Следующая команда возвращает ссылку на первую запись в цепочке:
chain_first(селектор, имя_цепочки, ссылка).

Отметим, что ссылка – это переменная адресного типа. По данной ссылке можно получить и саму запись.

11) Получение ссылки на следующую запись выполняет команда:
chain_next(селектор, ссылка_текущая, ссылка_следующая).
Заметим, что текущая ссылка должна быть задана.

12) Следующий предикат отыскивает и возвращает ссылку на указанную запись
chain_terms(селектор, имя_цепочки, тип_цепочки, запись, ссылка).

Пример программы.

```
DOMAINS
 db_selector=mydba
 dbdom = city(symbol, symbol)

 GOAL

db_open(mydba, ”d:\\work\\city.dat”,in_file),
db_chains(mydba, Chain),
chain_terms(mydba, Chain,dbdom,city(”London”,”England”),Ref),
 write(“REF=”,Ref).
```

Обратим внимание, что при указании маршрута к файлу БД наклонные черты пишутся парами. Приведенный фрагмент программы отыскивает ссылку на запись . **city(”London”,”England”)** и при удачном исходе поиска печатает ее на экране (в режиме DOS).

13) Следующая команда добавляет запись в начало цепочки
chain_inserta(селектор, имя_цепочки,тип_записи,запись,ссылка).
Последний аргумент не должен быть задан и получает значение. Пример:
chain_inserta(mydba, chain1,dbdom,city(”Minsk”,”Belarus”),Ref).

chain_insertz(селектор, имя_цепочки,тип_записи,запись,ссылка)
вставляет запись в конец цепочки, а команда

chain_insertafter(селектор, тип_записи,ссылка,запись, ссылка_новая)

вставляет **запись** после записи с указанной **ссылкой** и возвращает новую ссылку на вставленную запись. Последний аргумент не должен иметь значения.

14) Последняя команда данной группы удаляет всю цепочку:
chain_delete(селектор, имя_цепочки).

Рассматриваем теперь команды третьей группы.

15) Следующая команда удаляет запись из цепочки по ее ссылке:
term_delete(селектор, имя_цепочки, ссылка).

16) Следующая команда заменяет текущую запись, определяемую ссылкой, на новую и имеет такой формат
term_replace(селектор, тип_записи, ссылка, новая_запись).

17) Следующая команда (обратите на нее особое внимание) возвращает саму запись по ее ссылочному номеру:

ref_term(селектор, тип_записи, ссылка, запись)

Остается рассмотреть последнюю четвертую группу команд для работы с B-деревьями, ускоряющими поиск данных.

18) Команда создания дерева имеет такой формат:
bt_create(селектор, имя_дерева, селектор_дерева, длина_ключа, степень_ветвления_вершин)

Здесь **селектор_дерева** не должен быть задан и получает значение в результате выполнения команды. Остальные параметры должны быть заданы. **Длина_ключа** имеет тип UNSIGNED и определяет максимальное число символов в ключе; параметр **степень_ветвления_вершин** имеет тип UNSIGNED, и рекомендуется устанавливать его равным 3 или 4.

19) Команда **bt_close**(селектор, селектор_дерева) закрывает дерево.
Пример программы.

```
DOMAINS

 Db_selector= mydba

GOAL

  db_open(mydba, “dd.dat”,in_file),
  bt_create(mydba,”bt_cities”, Btree_Sel,10,4),
  %  какие-то действия
   bt_close(mydba,Btree_Sel),
   db_close(mydba).
```

20) Следующая команда удаляет B-дерево:
bt_delete(селектор, имя_дерева).

21) Для открытия B-дерева используется команда:

bt_open(селектор, имя_дерева, селектор_дерева).

Эта команда использует в качестве входных аргументов первые два. Селектор дерева возвращается в качестве результата.

22) Команда закрытия дерева имеет такой вид:
bt_close(селектор, селектор_дерева).

Основными операциями на дереве являются операции с ключами. Для вставки ключа в дерево используется команда:

23) **key_insert**(селектор, селектор_дерева, ключ, ссылка). Все аргументы должны быть заданы. Это значит, что запись уже должна содержаться в базе данных и на нее существует ссылка. Ссылку, в частности, можно получить при вставке записи в базу данных командой chain_inserta.
24) Удаление ключа из дерева выполняет команда

key_delete(селектор, селектор_дерева, ключ, ссылка) . Опять же все аргументы должны быть заданы.

25) Для получения ссылки по значению ключа используется команда:
key_search(селектор, селектор_дерева, ключ, ссылка). Первые три аргумента должны быть заданы, последний возвращается системой. Если в дереве нет указанного ключа, то возникает состояние неудачи и выполняется возврат к предыдущей точке ветвления.

26) Следующая команда возвращает значение ключа и ссылки для текущей записи:
key_current(селектор, селектор_дерева, ключ, ссылка). Первые два параметра должны быть заданы.
27) Наконец, последней командой рассматриваемой группы является команда
key_first(селектор, селектор_дерева, ссылка) .

Эта команда возвращает ссылку на первый ключ в дереве, хранящийся в его вершине.

Итак, мы рассмотрели большую часть команд для работы с внешними базами данных. Перейдем к описанию практической части работы.

В практической части работы необходимо построить окно с меню для работы с внешней базой данных. Каждая группа выбирает по своему усмотрению профиль базы данных: кадры, спорт, медицина, склад, товары и пр. Меню должно содержать следующие пункты:

1. Создать/Открыть/Удалить БД.
2. Ввод данных
3. Удаление данных
4. Поиск данных

Приведем пример реализации этих пунктов. Для создания БД будем использовать фиксированное имя “lab3.bin”. Профиль БД – футбольные команды. Определяем тип записей:

```
DOMAINS
    name=symbol
    country=symbol
```

```
    rate=integer
    teams=teams(name, country, rate)

  %  db_selector=mydb       объявлять не надо ! т.к. это объявление нужно
  %                          вставить в подсоединяемый файл  .pre

  CLAUSES
  create:-
      existfile("lab3.bin"),
      db_create(mydb,"lab3.bin",in_file),
      bt_create(mydb,"treeteem",Bsel,10,3),
      dlg_MessageBox("Info", "Created", mesbox_iconInformation,
                              mesbox_buttonsOK,
                              mesbox_defaultFirst, mesbox_suspendApplication),
       bt_close(mydb,Bsel),
       db_close(mydb).

   create:-
dlg_MessageBox("Info","SuchFilealreadyexists",mesbox_iconInformation,
mesbox_buttonsOK, mesbox_defaultFirst, mesbox_suspendApplication).
```

Прежде всего заметим, что объявление **db_selector=mydb** не надо делать внутри программы, так как оно подключается в программу через инструкцию
#include lab1.inc .Поэтому откройте файл lab1.inc и установите инструкцию нужным образом:

```
global domains
 DB_SELECTOR = browselist_db;mydb    % For treebrowser tool
```

Файл LAB1.INC создает сама система, используя имя приложения; в данном случае имя приложения – LAB1.
В этом примере задействована команда dlg_MessageBox. Первый параметр – название выводимого окна. Второй – содержание сообщения в окне. Третий – тип значка. Четвертый – отображаемые кнопки. Пятый-значение кнопки, возвращаемое по умолчанию. Шестой – тип действий , выполняемых до нажатия на кнопку. Приведенный фрагмент создает пустую базу данных.

Рассмотрим теперь фрагмент добавления записи в открытую базу данных. Данные для записи следует брать из полей редактирования, размещенных в окне. Вот фрагмент программы, который это делает:

```
%BEGIN mywin, id_add_rec
 win_mywin_eh(_Win,e_Menu(id_add_rec,_ShiftCtlAlt),0):-!,
   WinHandle1=win_GetCtlHandle(_Win,id_edit1),
   Name=win_GetText (WinHandle1),
   WinHandle2=win_GetCtlHandle(_Win,id_edit2),
   Country=win_GetText (WinHandle2),
   WinHandle3=win_GetCtlHandle(_Win,id_edit3),
   SRate=win_GetText (WinHandle3),
   str_int(Srate, Rate),
   chain_inserta(mydb,"chteems",teams,teams(Name,Country,Rate),Ref),
   bt_open(mydb,"Treeteams",BSel),
   key_insert(mydb, Bsel,Name,Ref),
```

```
bt_close(mydb,BSel), !.
```

Снова заметим, что добавление записей может производиться только в открытую базу. Вообще говоря, необходимо предусмотреть проверку, что база открыта в момент добавления записей. В этой работе мы, однако, оставим этот вопрос без ответа, предполагая, что пользователь обеспечит правильную последовательность операций над базой данных. **ВНИМАНИЕ:** прежде всего обеспечьте пункт меню для закрытия БД, поскольку при выходе из приложения с окрытой базой последняя помечается как INVALID. Такую базу можно открыть только с помощью команды **openInvalid**.

Последнее, что будет здесь рассмотрено – это операция поиска. Для программирования этой операции нужно ввести ключ поиска в поле редактирования. Ключом в нашем примере является название команды в поле Edit1. Фрагмент программного кода для обработки запроса на поиск приведен ниже.

```
%BEGIN mywin, idr_find
  win_mywin_eh(_Win,e_Menu(idr_find,_ShiftCtlAlt),0):-!,
   WinHandle1=win_GetCtlHandle(_Win,id_edit1),
    Name=win_GetText (WinHandle1),
     bt_open(mydblab, "Treeteams", BSel),
     key_search(mydb, Bsel,Name,Ref),
     ref_term(mydb, teams, Ref, Term),
     Term=teams(_, Country, Rate),
     WinHandle2=win_GetCtlHandle(_Win,id_edit2),
     win_SetText (WinHandle2,Country),
     WinHandle3=win_GetCtlHandle(_Win,id_edit3),
      str_int(Srate, Rate),
      win_SetText (WinHandle3, Srate),
      bt_close(mydb, BSel),
          !.
%END mywin, idr_find
```

Итак, мы рассмотрели основные пункты по реализации управления БД. В последующем БД будет использоваться для отработки стратегии вывода на основании теоремы Байеса. Поэтому данную работу следует выполнять особенно тщательно.

КОНТРОЛЬНЫЕ ВОПРОСЫ.

1. Что такое внешняя база данных?
2. Какие группы команд для работы с внешними БД вам известны?
3. Назовите основные команды из групп db_... ,
4. Назовите основные команды из группы chain ...
5. Назовите основные команды из группы term ...
6. Назовите основные команды из группы bt_ , key_..
7. Объясните, когда база данных получает статус INVALID ?
8. Объясните работу вашей программы

ВАРИАНТЫ ЗАДАНИЙ

1. Создать базу данных и интерфейс для работы с ней по теме «Студенты»

2. Создать базу данных и интерфейс для работы с ней по теме «Товары»
3. Создать базу данных и интерфейс для работы с ней по теме «Книги»
4. Создать базу данных и интерфейс для работы с ней по теме «Автомобили»
5. Создать базу данных и интерфейс для работы с ней по теме «Турпоездки»
6. Создать базу данных и интерфейс для работы с ней по теме «Футбол»
7. Создать базу данных и интерфейс для работы с ней по теме «Метро»