ЛАБОРАТОРНАЯ РАБОТА № 1
«СОЗДАНИЕ ДИНАМИЧЕСКИХ БАЗ ДАННЫХ»

Цель работы

Изучение технологии подготовки и выполнения Пролог-программ в интегрированной среде, исследование способов организации динамических баз данных (БД) средствами языка Пролог.

Постановка задачи

1. Изучить среду программирования Eclipse ProDT по методическим указаниям [3]. Выполнить все приведенные примеры в окне консоли среды программирования.

2. Ознакомиться по лекционному материалу или книгам [1,2] с объектами данных и сопоставлением в языке Пролог, организацией управления в пролог-программах, предикатами обработки списков, встроенными предикатами работы с базой данных и предикатами ввода-вывода. Изучить примеры применения этих предикатов, приведенные в разделе 1.2 настоящей лабораторной работы.

2. Ознакомиться с вариантом задания и выбрать один из способов для хранения записей базы данных (см. п. 1.2.5).

3. Ознакомиться с примером кода, приведенного в приложении А, и по аналогии определить на языке Пролог для заданного варианта предикаты добавления записи в базу, удаления записи, просмотра базы, сохранения базы в файле и загрузки её из файла.

4. Создать в среде программирования Пролог-проект в соответствии с методическими указаниями [3], содержащий подготовленные определения предикатов, указанных в п. 1.4.3.

5. Выполнить частичную отладку проекта.

6. Разработать определения дополнительных предикатов выборки и корректировки записей базы данных в соответствии с вариантом. При этом выборку записей из базы выполнять путем реализации операции проекции реляционной алгебры, следуя общим рекомендациям, указанным в п. 1.2.7.

7. Разработать предикаты, реализующие примеры операций реляционной алгебры (объединение, пересечение, разность) в соответствии с п. 1.2.7.

8. Выполнить полную отладку проекта и зафиксировать результаты работы программы в виде экранных копий.

Вариант 3

Написать программу, обеспечивающую создание динамической базы данных. Структура базы данных определяется одной из таблиц в соответствии с вариантом задания. В функции программы должно входить:

− добавление записи в базу данных;

− удаление записи из базы данных;

− просмотр базы данных;

− сохранение базы данных в файле;

− загрузка базы данных из файла;

− реализация операций реляционной алгебры (на примерах).

Кроме этого, программа должна выполнять дополнительные функции, указанные в варианте задания (таблица 1).

Таблица 1 – Вариант задания

Вариант	Номер таблицы и дополнительные функции
3.	Таблица 2. Корректировка данных в базе по номеру группы; вывод на дисплей фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2; если таких студентов нет, вывести соответствующее сообщение.

Таблица 2 – Студент группы

Фамилия И. О.	Номер группы	Успеваемость				
		P1	P2	P3	P4	P5

Ход работы

1. Была написана программа, обеспечивающая создание динамической базы данных. Сперва была добавлена возможность добавления записи в базу данных, соответствующее правило представлено в листинге 1.

Листинг 1 – Правило добавления студента в базу данных

```
add_student:-
    tty_clear,
    write('Введите Фамилия И.О.'),
    read(FIO),
    write('Введите номер группы'),
    read(Group),
    write('Введите успеваемость по предмету P1'),
    read(P1),
    write('Введите успеваемость по предмету P2'),
    read(P2),
    write('Введите успеваемость по предмету P3'),
    read(P3),
    write('Введите успеваемость по предмету P4'),
    read(P4),
    write('Введите успеваемость по предмету P5'),
    read(P5),
    assert(student(FIO, Group, P1, P2, P3, P4, P5)).
```

2. Затем была добавлена возможность удаления записи из базы данных, соответствующее правило представлено в листинге 2.

Листинг 2 – Правило удаления студента из базы данных

```
delete_student:-
    tty_clear,
    write('Введите номер группы студента для удаления (удалятся все студенты
данной группы)'),read(Group),
    retract(student(_,Group,_,_,_,_,_)),
    write('Все студенты группы '),write(Group),write(' были успешно
удалены'),nl,
    get(temp),!;
    write('Студентов данной группы не найдено, возврат в меню'),
```

```
        get(temp).
```

3. После чего было добавлено правило, позволяющее вывести список всех записей в базе данных, код представлен в листинге 3.

Листинг 3 – Правило для вывода списка студентов

```
show_students:-
    tty_clear,
    student(FIO, Group, P1, P2, P3, P4, P5),
   write('Фамилия И.О.         Номер группы         P1         P2         P3
P4         P5'),nl,

write(FIO),tab(10),write(Group),tab(10),write(P1),tab(10),write(P2),tab(10),write(
P3),tab(10),write(P4),tab(10),write(P5),nl,

write('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~'),nl,
    fail;
    true,
    get(temp).
```

4. Затем были добавлены правила для сохранения и загрузки базы данных в файл, код представлен в листинге 4 и листинге 5 соответственно.

Листинг 4 – Правило для сохранения базы данных в файл

```
save_file:-
    tty_clear,
    tell('students.txt'),
    listing(student(_,_,_,_,_,_,_)),
    told(),
    write('Таблица студентов была успешно сохранена в файл
students.txt'),nl,get(temp).
```

Листинг 5 – Правило для загрузки базы данных из файла

```
load_file:-
    tty_clear,
    retractall(student(_,_,_,_,_,_,_)),
    consult('students.txt'),
    listing(student(_,_,_,_,_,_,_)),
    write('Таблица студентов была успешно загружена из файла
students.txt'),nl,get(temp).
```

5. Затем были добавлены правила для выполнения операций реляционной алгебры, представленные в листинге 6.

Листинг 6 – Правила для выполнения операций реляционной алгебры

```
relational_algebra:-
    tty_clear,
    write('Формирование отношения r1: студенты группы "ИС/Б-20-2-о"'),nl,
    student_subset("ИС/Б-20-2-о", R1),
    list_to_db(R1),
    print_list(R1),
    write('Формирование отношения r2: студенты группы "ИС/Б-20-1-о"'),nl,
    student_subset("ИС/Б-20-1-о", R2),
    list_to_db(R2),
    print_list(R2),
    write('Объединенное отношение r1_или_r2'),nl,
    union(Rez1),
    print_list(Rez1),
    write('Пересечение отношерний r1_и_r2'),nl,
    intersect(Rez2),
    print_list(Rez2),
    write('Разность отношений r1_и_не_r2'),nl,
    minus(Rez3),
    print_list(Rez3),

    get(temp).

student_subset(Group, R):-
    bagof(student_f(FIO,Group,P1,P2,P3,P4,P5),
        student(FIO,Group,P1,P2,P3,P4,P5),R).

list_to_db([]).
list_to_db([H|T]):-
    H=student_f(FIO,Group,P1,P2,P3,P4,P5),
    assertz(student_f(FIO,Group,P1,P2,P3,P4,P5)),
    list_to_db(T).

print_list([]).
print_list([H|T]):-
    write(H),nl,print_list(T).

union_r1_r2(X1,X2,X3,X4,X5,X6,X7):-
    student_f(X1,"ИС/Б-20-2-о",X3,X4,X5,X6,X7),X2="ИС/Б-20-2-о";
    student_f(X1,"ИС/Б-20-1-о",X3,X4,X5,X6,X7),X2="ИС/Б-20-1-о".

union(Rez):-
    bagof(student_f1_or_f2(X1,X2,X3,X4,X5,X6,X7),
        union_r1_r2(X1,X2,X3,X4,X5,X6,X7),Rez).

intersect_r1_r2(X11,X12,X13,X14,X15,X16,X17,X22,X23,X24,X25,X26,X27):-
    student_f(X11,"ИС/Б-20-1-о",X13,X14,X15,X16,X17),X12="ИС/Б-20-1-о",
    student_f(X11,"ИС/Б-20-2-о",X23,X24,X25,X26,X27),X22="ИС/Б-20-2-о".

intersect(Rez):-

bagof(student_f1_and_f2(X11,X12,X13,X14,X15,X16,X17,X22,X23,X24,X25,X26,X27),

intersect_r1_r2(X11,X12,X13,X14,X15,X16,X17,X22,X23,X24,X25,X26,X27),Rez).

minus_r1_r2(X11,X12,X13,X14,X15,X16,X17):-
```

```
        student_f(X11,"ИС/Б-20-2-о",X13,X14,X15,X16,X17),X12="ИС/Б-20-2-о",
        not(student_f(X11,"ИС/Б-20-1-о",X23,X24,X25,X26,X27)),X22="ИС/Б-20-1-о".

    minus(Rez):-
        bagof(student_f1_and_not_f2(X11,X12,X13,X14,X15,X16,X17),
            minus_r1_r2(X11,X12,X13,X14,X15,X16,X17),Rez).
```

6. Далее было добавлено правило для корректировки данных в базе по номеру группы, которое представлено в листинге 7.

Листинг 7 – Правило для изменения студента по номеру группы

```
    edit_student:-
        tty_clear,
       write('Введите номер группы студента для изменения (изменяться будет первый
встреченный студент выбранной группы)'),nl,read(EditGroup),
        retract(student(_,EditGroup,_,_,_,_,_)),
        write('Введите Фамилия И.О.'),
        read(FIO),
        write('Введите номер группы'),
        read(Group),
        write('Введите успеваемость по предмету P1'),
        read(P1),
        write('Введите успеваемость по предмету P2'),
        read(P2),
        write('Введите успеваемость по предмету P3'),
        read(P3),
        write('Введите успеваемость по предмету P4'),
        read(P4),
        write('Введите успеваемость по предмету P5'),
        read(P5),
        assert(student(FIO, Group, P1, P2, P3, P4, P5)).
```

7. Затем было добавлено правило, при помощи которого можно вывести всех студентов, у которых есть хотя бы одна оценка 2, в противном случае выводится сообщение о том, что таких студентов нет, код представлен в листинге 8.

Листинг 8 – Правило для вывода студентов хотя бы с одной оценкой 2

```
    show_dummies:-
        tty_clear,
        student(FIO, Group, P1, P2, P3, P4, P5),
        member(2,[P1,P2,P3,P4,P5]),
       write('Фамилия И.О.         Номер группы         P1         P2         P3
P4         P5'),nl,

write(FIO),tab(10),write(Group),tab(10),write(P1),tab(10),write(P2),tab(10),write(
P3),tab(10),write(P4),tab(10),write(P5),nl,

write('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~'),nl,get(temp),fail,!;
```

```
write('Больше нет студентов с оценками 2'),get(temp).
```

Код полной программы на языке swi prolog представлен в листине А1 приложения А.

Выводы

В ходе выполнения лабораторной работы были изучены технологии подготовки и выполнения Пролог-программ в интегрированной среде, исследованы способы организации динамических баз данных (БД) средствами языка Пролог.

ПРИЛОЖЕНИЕ А

Код программы для работы с динамической базой данных студентов

Листинга А1 – Код программы

```
:- dynamic student/7, student_f/7.

student("Иванов И. В", "ИС/Б-20-1-о", 3, 4, 5, 4, 4).
student("Иванов И. В", "ИС/Б-20-2-о", 4, 3, 5, 3, 4).
student("Петров И. В", "ИС/Б-20-2-о", 5, 3, 3, 3, 4).
student("Сидоров И. В", "ИС/Б-20-3-о", 3, 4, 5, 4, 4).
student("Васечкин И. В", "ИС/Б-20-2-о", 3, 4, 5, 4, 3).

% ~ RULES TO CONTROL USER INPUT THROUGH MENU

main_menu:-
    tty_clear,
    write('1 - Добавление записи в базу данных'),nl,
    write('2 - Удаление записи из базы данных'),nl,
    write('3 - Просмотр базы данных'),nl,
    write('4 - Сохранение базы данных в файле'),nl,
    write('5 - Загрузка базы данных из файла'),nl,
    write('6 - Реализация операций реляционной алгебры'),nl,
    write('7 - Корректировка данных в базе по номеру группы'),nl,
    write('8 - Просмотр студентов, имеющих хотя бы одну оценку 2'),nl,
    write('9 - Выход'),nl,
    read(Item), main_menu_item(Item).

main_menu_item(1):-
    add_student,
    main_menu,!.

main_menu_item(2):-
    delete_student,
    main_menu,!.

main_menu_item(3):-
    show_students,
    main_menu,!.

main_menu_item(4):-
    save_file,
    main_menu,!.

main_menu_item(5):-
    load_file,
    main_menu,!.

main_menu_item(6):-
    relational_algebra,
    main_menu,!.

main_menu_item(7):-
    edit_student,
```

```
        main_menu,!.

    main_menu_item(8):-
        show_dummies,
        main_menu,!.

    main_menu_item(9):-tty_clear,!.

    main_menu_item(_Item):-
        tty_clear,
        write('Нет такого пункта меню, выберите пункт в пределах от 1 до 8'),nl,
        main_menu.

    % ~ RULE TO ADD STUDENT TO DB

    add_student:-
        tty_clear,
        write('Введите Фамилия И.О.'),
        read(FIO),
        write('Введите номер группы'),
        read(Group),
        write('Введите успеваемость по предмету P1'),
        read(P1),
        write('Введите успеваемость по предмету P2'),
        read(P2),
        write('Введите успеваемость по предмету P3'),
        read(P3),
        write('Введите успеваемость по предмету P4'),
        read(P4),
        write('Введите успеваемость по предмету P5'),
        read(P5),
        assert(student(FIO, Group, P1, P2, P3, P4, P5)).

    % ~ RULE TO SHOW STUDENTS DB

    show_students:-
        tty_clear,
        student(FIO, Group, P1, P2, P3, P4, P5),
       write('Фамилия И.О.         Номер группы         P1         P2         P3
P4         P5'),nl,

write(FIO),tab(10),write(Group),tab(10),write(P1),tab(10),write(P2),tab(10),write(
P3),tab(10),write(P4),tab(10),write(P5),nl,

write('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~'),nl,
        fail;
        true,
        get(temp).

    % ~ RULE TO DELETE STUDENTS OF EXACT GROUP

    delete_student:-
        tty_clear,
        write('Введите номер группы студента для удаления (удалятся все студенты
данной группы)'),read(Group),
        retract(student(_,Group,_,_,_,_,_)),
```

```
          write('Все студенты группы '),write(Group),write(' были успешно
удалены'),nl,
          get(temp),!;
          write('Студентов данной группы не найдено, возврат в меню'),
          get(temp).

      % ~ RULE TO SAVE DB TO FILE

      save_file:-
          tty_clear,
          tell('students.txt'),
          listing(student(_,_,_,_,_,_,_)),
          told(),
          write('Таблица студентов была успешно сохранена в файл
students.txt'),nl,get(temp).

      % ~ RULE TO LOAD DB FROM FILE

      load_file:-
          tty_clear,
          retractall(student(_,_,_,_,_,_,_)),
          consult('students.txt'),
          listing(student(_,_,_,_,_,_,_)),
          write('Таблица студентов была успешно загружена из файла
students.txt'),nl,get(temp).

      % ~ RULE TO SHOW STUDENTS WITH AT LEAST ONE '2' MARK

      show_dummies:-
          tty_clear,
          student(FIO, Group, P1, P2, P3, P4, P5),
          member(2,[P1,P2,P3,P4,P5]),
         write('Фамилия И.О.         Номер группы         P1         P2         P3
P4         P5'),nl,

write(FIO),tab(10),write(Group),tab(10),write(P1),tab(10),write(P2),tab(10),write(
P3),tab(10),write(P4),tab(10),write(P5),nl,

write('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~'),nl,get(temp),fail,!;
          write('Больше нет студентов с оценками 2'),get(temp).

      % ~ RULE TO EDIT STUDENT BY HIS GROUP NUMBER

      edit_student:-
          tty_clear,
         write('Введите номер группы студента для изменения (изменяться будет первый
встреченный студент выбранной группы)'),nl,read(EditGroup),
          retract(student(_,EditGroup,_,_,_,_,_)),
          write('Введите Фамилия И.О.'),
          read(FIO),
          write('Введите номер группы'),
          read(Group),
          write('Введите успеваемость по предмету P1'),
          read(P1),
          write('Введите успеваемость по предмету P2'),
          read(P2),
          write('Введите успеваемость по предмету P3'),
```

```
    read(P3),
    write('Введите успеваемость по предмету P4'),
    read(P4),
    write('Введите успеваемость по предмету P5'),
    read(P5),
    assert(student(FIO, Group, P1, P2, P3, P4, P5)).

% ~ RELATIONAL ALGEBRA RULES

relational_algebra:-
    tty_clear,
    write('Формирование отношения r1: студенты группы "ИС/Б-20-2-о"'),nl,
    student_subset("ИС/Б-20-2-о", R1),
    list_to_db(R1),
    print_list(R1),
    write('Формирование отношения r2: студенты группы "ИС/Б-20-1-о"'),nl,
    student_subset("ИС/Б-20-1-о", R2),
    list_to_db(R2),
    print_list(R2),
    write('Объединенное отношение r1_или_r2'),nl,
    union(Rez1),
    print_list(Rez1),
    write('Пересечение отношерний r1_и_r2'),nl,
    intersect(Rez2),
    print_list(Rez2),
    write('Разность отношений r1_и_не_r2'),nl,
    minus(Rez3),
    print_list(Rez3),

    get(temp).

student_subset(Group, R):-
    bagof(student_f(FIO,Group,P1,P2,P3,P4,P5),
        student(FIO,Group,P1,P2,P3,P4,P5),R).

list_to_db([]).
list_to_db([H|T]):-
    H=student_f(FIO,Group,P1,P2,P3,P4,P5),
    assertz(student_f(FIO,Group,P1,P2,P3,P4,P5)),
    list_to_db(T).

print_list([]).
print_list([H|T]):-
    write(H),nl,print_list(T).

union_r1_r2(X1,X2,X3,X4,X5,X6,X7):-
    student_f(X1,"ИС/Б-20-2-о",X3,X4,X5,X6,X7),X2="ИС/Б-20-2-о";
    student_f(X1,"ИС/Б-20-1-о",X3,X4,X5,X6,X7),X2="ИС/Б-20-1-о".

union(Rez):-
    bagof(student_f1_or_f2(X1,X2,X3,X4,X5,X6,X7),
        union_r1_r2(X1,X2,X3,X4,X5,X6,X7),Rez).

intersect_r1_r2(X11,X12,X13,X14,X15,X16,X17,X22,X23,X24,X25,X26,X27):-
    student_f(X11,"ИС/Б-20-1-о",X13,X14,X15,X16,X17),X12="ИС/Б-20-1-о",
    student_f(X11,"ИС/Б-20-2-о",X23,X24,X25,X26,X27),X22="ИС/Б-20-2-о".

intersect(Rez):-
```

```
bagof(student_f1_and_f2(X11,X12,X13,X14,X15,X16,X17,X22,X23,X24,X25,X26,X27),

intersect_r1_r2(X11,X12,X13,X14,X15,X16,X17,X22,X23,X24,X25,X26,X27),Rez).

      minus_r1_r2(X11,X12,X13,X14,X15,X16,X17):-
          student_f(X11,"ИС/Б-20-2-о",X13,X14,X15,X16,X17),X12="ИС/Б-20-2-о",
          not(student_f(X11,"ИС/Б-20-1-о",X23,X24,X25,X26,X27)),X22="ИС/Б-20-1-о".

      minus(Rez):-
          bagof(student_f1_and_not_f2(X11,X12,X13,X14,X15,X16,X17),
              minus_r1_r2(X11,X12,X13,X14,X15,X16,X17),Rez).

      :- main_menu.
```