**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Кафедра теоретической и прикладной информатики**

**Лабораторная работа №** 4

**по дисциплине**  Базы данных и экспертные системы

вариант задания № 2

**Факультет: ФПМИ**

**Группа: ПМИ - 72**

**Бригада №7**

**Студенты: Антонов С.**

**Боброва А.**

**Преподаватели: Стасышина Т.Л.**

**Сивак М.А.**

**Новосибирск 2020**

**Задание**

1. Изучить общие правила подготовки программ и использования

программных средств ESQL/C, правила использования главных переменных, средства встраивания SQL-описаний в С-программы, структуру SQLCA, назначение и структуру заголовочных файлов.

2. Разработать и отладить ESQL/С-программу, реализующую задачу 1 из соответствующего варианта заданий, результатом которой является единственная строка.

3. Разработать и отладить ESQL/С-программу, реализующую задачу 2 из соответствующего варианта заданий и связанную с модификацией базы данных.

4. Изучить синтаксис и правила использования операторов Declare,

Open, Fetch, Close, а также особенности работы с курсором.

5. Разработать и отладить набор ESQL/С-программ, решающих задачи 3–5 из соответствующего варианта заданий с использованием аппарата курсоров (последовательного и скроллирующего). Результатом

работы программ является набор строк, которые подлежат выводу на

экран с соответствующими пояснительными заголовками.

**Ход работы**

1. Выдать число изделий, для которых детали с весом больше 12 поставлял первый по алфавиту поставщик.

select count(distinct spj.n\_izd) kol\_izd

from spj

join p on p.n\_det=spj.n\_det

join s on s.n\_post=spj.n\_post

where p.ves>12

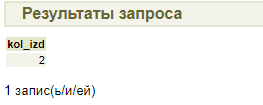
and

s.name = (select name

from s

order by 1

limit 1)



2. Поменять местами фамилии первого и последнего по алфавиту поставщика, т. е. первому по алфавиту поставщику установить фамилию последнего по алфавиту поставщика и наоборот.

update s set name = (case when s.name=(select s5.name

from s s5

order by 1

limit 1)

then (select s3.name name1

from s s3

order by 1 desc

limit 1 )

else (select s4.name name2

from s s4

order by 1

limit 1)

end )

where s.name=(select name from s order by 1 limit 1)

or

s.name=(select name from s order by 1 desc limit 1)

До модификации:



После модификации:



3. Найти изделия, для которых выполнены поставки, вес которых более чем в 4 раза превышает минимальный вес поставки для изделия. Вывести номер изделия, вес поставки, минимальный вес поставки для изделия.

select a.n\_izd, a.kol\*pa.ves pves, b.mves

from spj a

join p pa on pa.n\_det=a.n\_det

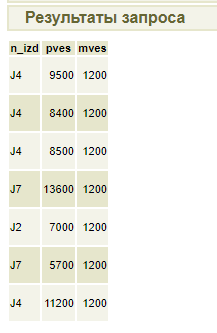
join (select t.n\_izd, min(t.kol\*p.ves)mves

from spj t

join p on p.n\_det=t.n\_det

group by t.n\_izd) b on b.n\_izd=a.n\_izd

where a.kol\*pa.ves > 4\*mves



4. Выбрать поставщиков, не поставивших ни одной из деталей, имеющих наименьший вес.

select s.n\_post

from s

except

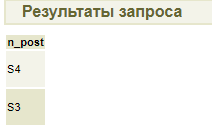
select spj.n\_post

from spj

where spj.n\_det in (select p.n\_det

from p

where p.ves = (select min(p1.ves) from p p1))



5. Выдать полную информацию о поставщиках, поставляющих ТОЛЬКО красные детали и только для изделия с длиной названия не меньше 7.

select distinct s.\*

from spj t

join s on s.n\_post=t.n\_post

where t.n\_post in (select n\_post

from spj

join p on p.n\_det=spj.n\_det

join j on j.n\_izd=spj.n\_izd

where p.cvet = 'Красный'

and

length(j.name)>7)

except

select distinct s.\*

from spj t

join s on s.n\_post=t.n\_post

where not t.n\_post in (select n\_post

from spj

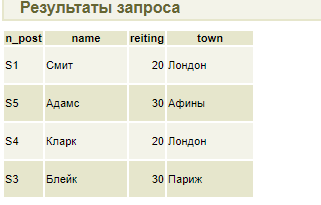
join p on p.n\_det=spj.n\_det

join j on j.n\_izd=spj.n\_izd

where p.cvet = 'Красный'

and

length(j.name)>7 )



**Текст программы:**

#include <sqlca.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

bool Connect()

{

EXEC SQL connect to students@fpm2.ami.nstu.ru user "pmi-b7607" using "apjafOd0";

if (sqlca.sqlcode != 0)

{

do\_error("Connect to database", sqlca.sqlcode);

return;

}

EXEC SQL set search\_path to pmib7607;

if (sqlca.sqlcode != 0)

{

do\_error("Set database", sqlca.sqlcode);

return;

}

}

void do\_error(char\* st\_name, int errnum)

{

printf("Error code %d at %s \n", errnum, st\_name);

printf("Error message: %s\n", sqlca.sqlerrm.sqlerrmc);

return;

}

int Task1()

{

EXEC SQL begin declare section;

int n;

EXEC SQL end declare section;

printf("Task1\n");

EXEC SQL begin work;

EXEC SQL select count(distinct spj.n\_izd) into:n

from spj

join p on p.n\_det=spj.n\_det

join s on s.n\_post=spj.n\_post

where p.ves>12

and

s.name = (select name

from s

order by 1

limit 1);

if (sqlca.sqlcode != 0)

{

do\_error("Request", sqlca.sqlcode);

EXEC SQL rollback work;

return 1;

}

printf("Kol\_det: %d\n", n);

EXEC SQL commit work;

return 0;

}

int Task2()

{

printf("Task2\n");

EXEC SQL begin work;

EXEC SQL update s set name = (case when s.name = (select s5.name

from s s5

order by 1

limit 1)

then(select s3.name name1

from s s3

order by 1 desc

limit 1)

else (select s4.name name2

from s s4

order by 1

limit 1)

end)

where s.name = (select name from s order by 1 limit 1)

or

s.name = (select name from s order by 1 desc limit 1);

if (sqlca.sqlcode != 0)

{

do\_error("Request", sqlca.sqlcode);

EXEC SQL rollback work;

return 1;

}

printf("Ok\n");

printf("%ld lines processed\n", sqlca.sqlerrd[2]);

EXEC SQL commit work;

return 0;

}

int Task3()

{

EXEC SQL begin declare section;

char \_n\_izd[12];

int \_kol;

float \_mves;

EXEC SQL end declare section;

printf("Task3\n");

EXEC SQL DECLARE cur3 CURSOR FOR

select a.n\_izd, a.kol\* pa.ves pves, b.mves

from spj a

join p pa on pa.n\_det = a.n\_det

join(select t.n\_izd, min(t.kol \* p.ves)mves

from spj t

join p on p.n\_det = t.n\_det

group by t.n\_izd) b on b.n\_izd = a.n\_izd

where a.kol \* pa.ves > 4 \* mves;

if (sqlca.sqlcode != 0)

{

do\_error("Declare cursor", sqlca.sqlcode);

return 1;

}

EXEC SQL begin work;

EXEC SQL OPEN cur3;

if (sqlca.sqlcode != 0)

{

do\_error("Open cursor", sqlca.sqlcode);

EXEC SQL rollback work;

EXEC SQL close cur3;

return 1;

}

EXEC SQL FETCH NEXT FROM cur3 into : \_n\_izd, : \_kol, : \_mves;

if (sqlca.sqlcode == 100)

{

printf("No results for request\n");

EXEC SQL close cur3;

EXEC SQL commit work;

return 0;

}

else if (sqlca.sqlcode < 0)

{

do\_error("Fetch", sqlca.sqlcode);

EXEC SQL close cur3;

EXEC SQL rollback work;

return 1;

}

printf("n\_izd\tkol\tmves\n");

do

{

printf("%s\t\t%d\t\t%f\n", \_n\_izd, \_kol, \_mves);

EXEC SQL fetch next from cur3 into : \_n\_izd, : \_kol, : \_mves;

if (sqlca.sqlcode < 0)

{

do\_error("Fetch", sqlca.sqlcode);

EXEC SQL close cur3;

EXEC SQL rollback work;

return 1;

}

} while (sqlca.sqlcode != 100);

EXEC SQL CLOSE cur3;

if (sqlca.sqlcode != 0)

{

do\_error("Close cursor", sqlca.sqlcode);

return 1;

}

EXEC SQL commit work;

}

int Task4()

{

EXEC SQL begin declare section;

char n\_post[12];

EXEC SQL end declare section;

EXEC SQL declare cur4 cursor for

select s.n\_post

from s

except

select spj.n\_post

from spj

where spj.n\_det in(select p.n\_det

from p

where p.ves = (select min(p1.ves) from p p1));

if (sqlca.sqlcode != 0)

{

do\_error("Declare cursor", sqlca.sqlcode);

return 1;

}

EXEC SQL begin work;

EXEC SQL open cur4;

if (sqlca.sqlcode != 0)

{

do\_error("Open cursor", sqlca.sqlcode);

EXEC SQL rollback work;

EXEC SQL close cur4;

return 1;

}

EXEC SQL fetch next from cur4 into : n\_post;

if (sqlca.sqlcode == 100)

{

printf("No results for request\n");

EXEC SQL close cur4;

EXEC SQL commit work;

return 0;

}

else if (sqlca.sqlcode < 0)

{

do\_error("Fetch", sqlca.sqlcode);

EXEC SQL close cur4;

EXEC SQL rollback work;

return 1;

}

printf("n\_post\n");

do

{

printf("%s\n", n\_post);

EXEC SQL fetch next from cur4 into : n\_post;

if (sqlca.sqlcode < 0)

{

do\_error("Fetch", sqlca.sqlcode);

EXEC SQL close cur4;

EXEC SQL rollback work;

return 1;

}

} while (sqlca.sqlcode != 100);

EXEC SQL close cur4;

if (sqlca.sqlcode != 0)

{

do\_error("Close cursor", sqlca.sqlcode);

return 1;

}

EXEC SQL commit work;

return 0;

}

int Task5()

{

EXEC SQL begin declare section;

char n\_post[15], name1[20], town1[20];

int reiting;

EXEC SQL end declare section;

EXEC SQL declare cur5 cursor for

select distinct s.\*

from spj t

join s on s.n\_post = t.n\_post

where t.n\_post in(select n\_post

from spj

join p on p.n\_det = spj.n\_det

join j on j.n\_izd = spj.n\_izd

where p.cvet = 'РљСЂР°СЃРЅС‹Р№'

and

length(j.name) > 7)

except

select distinct s.\*

from spj t

join s on s.n\_post = t.n\_post

where not t.n\_post in(select n\_post

from spj

join p on p.n\_det = spj.n\_det

join j on j.n\_izd = spj.n\_izd

where p.cvet = 'РљСЂР°СЃРЅС‹Р№'

and

length(j.name) > 7);

if (sqlca.sqlcode != 0)

{

do\_error("Declare cursor", sqlca.sqlcode);

return 1;

}

EXEC SQL begin work;

EXEC SQL open cur5;

if (sqlca.sqlcode != 0)

{

do\_error("Open cursor", sqlca.sqlcode);

EXEC SQL rollback work;

EXEC SQL close cur5;

return 1;

}

EXEC SQL fetch next from cur5 into : n\_post, : name1, : reiting, : town1;

if (sqlca.sqlcode == 100)

{

printf("No results for request\n");

EXEC SQL close cur5;

EXEC SQL commit work;

return 0;

}

else if (sqlca.sqlcode < 0)

{

do\_error("Fetch", sqlca.sqlcode);

EXEC SQL close cur5;

EXEC SQL rollback work;

return 1;

}

printf("n\_post\tname\t\treiting \t\ttown\n");

do

{

printf("%s\t%s\t%d\t%s\n", n\_post, name1, reiting, town1);

EXEC SQL fetch next from cur5 into : n\_post, : name1, : reiting, : town1;

if (sqlca.sqlcode < 0)

{

do\_error("Fetch", sqlca.sqlcode);

EXEC SQL close cur5;

EXEC SQL rollback work;

return 1;

}

} while (sqlca.sqlcode != 100);

EXEC SQL close cur5;

if (sqlca.sqlcode != 0)

{

do\_error("Close cursor", sqlca.sqlcode);

return 1;

}

EXEC SQL commit work;

}

void main()

{

setlocale(LC\_ALL, "Russian");

printf("Start...\n");

if (Connect())

{

printf("Error connect\n");

return 1;

}

printf("Connect...\n");

int n, flag = 0;

do {

printf("Enter number:1-6:\n");

scanf("%d", &n);

switch (n)

{

case 1:

Task1();

break;

case 2:

Task2();

break;

case 3:

Task3();

break;

case 4:

Task4();

break;

case 5:

Task5();

break;

case 6:

flag = 1;

EXEC SQL disconnect all;

printf("Session closed\n");

break;

default:

printf("Wrong number!\n");

break;

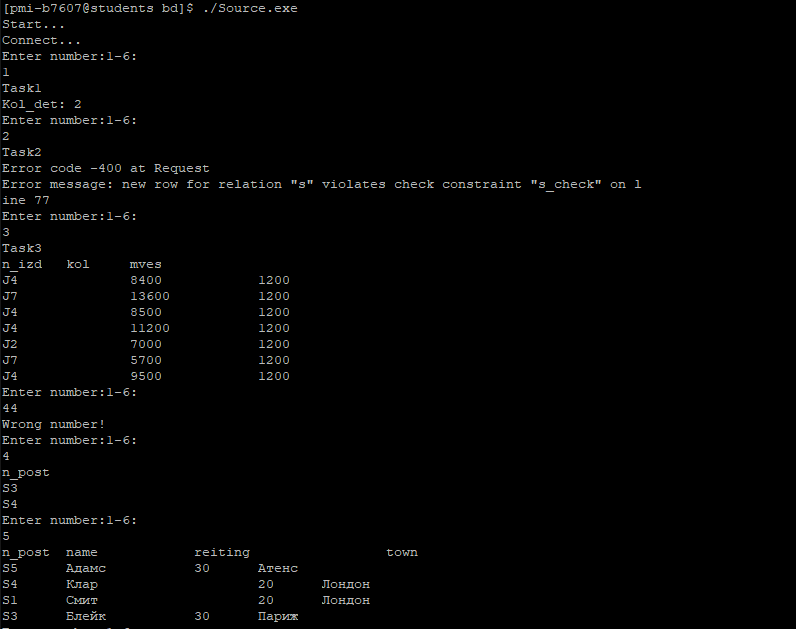
}

} while (flag == 0);

return;

}

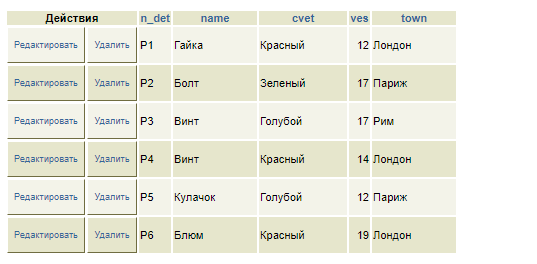
**Работа программы**



Тесты ошибок:

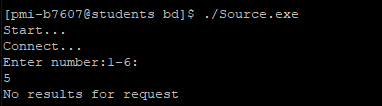
Изменим значения в таблице так, чтобы один из запросов ничего не вернул: уберем все детали красного цвета и проверим пятый запрос

ДО

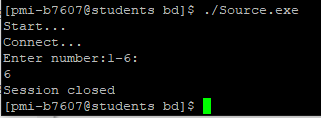


ПОСЛЕ





Корректное завершение сессии:



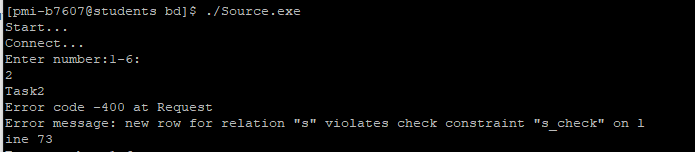
Проверка работы программы при сбое:

Зададим для таблицы s ограничение check , которое будет на одних данных выполняться, а на других - нет. И получим сообщение, что ограничение нарушено.

Ограничение:

CHECK (length(name)+length(town)<11)

Т.к сумма первого и последнего по алфавиту равна 10 , но имя первого на одну букву длиннее, то при их перестановке появится ошибка.



**Запрос 1 по шагам**

***Выдать число изделий, для которых детали с весом больше 12 поставлял первый по алфавиту поставщик.***

**Анализ задания:**

Есть поставщик первый по алфавиту.

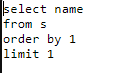
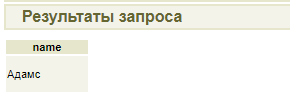
Вес деталей , которые поставляются для изделий.

Нужно выбрать поставки такие, что поставляемая деталь подходит по весу из указанного множества.

Подсчитать число разных изделий в полученной выборке поставок.

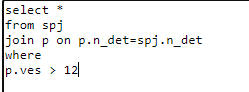
**Подзадача 1**

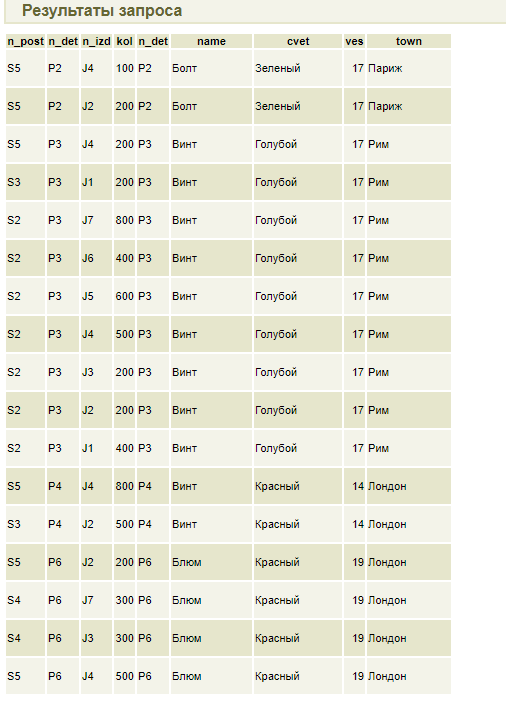
*Найти первого по алфавиту поставщика*

**Подзадача 2**

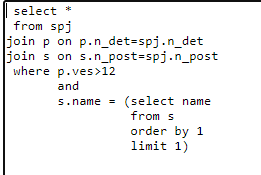
*Найти постави для которых поставлялись детали с весом больше 12*

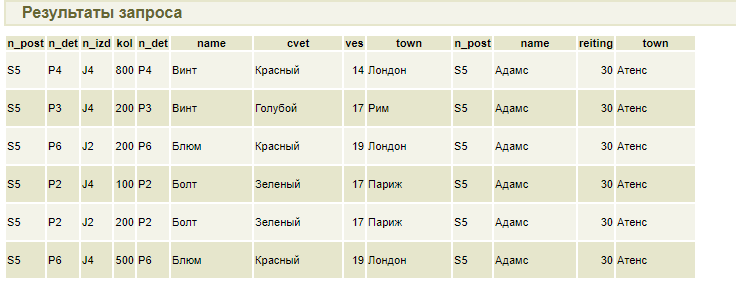




**Подзадача 3**

*Выбрать такие поставки, что деталь поставки весом больше 12 поставлена поставшиком из предыдущего запроса*





**Окончательный запрос**

***Выдать число изделий, для которых детали с весом больше 12 поставлял первый по алфавиту поставщик***

