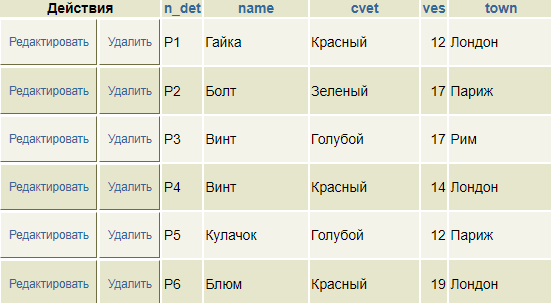
|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
|  | | |
| Лабораторная работа № 3 | | |
| по дисциплине «Технологии баз данных» | | |
|  | | |
| Вариант задания №2 | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-02 |
| Бригада: | 6 |
| Студент: | Сидоров Даниил, |
|  | Дюков Богдан |
| Преподаватель: | Стасышина Т.Л., |
|  | Сивак М.А. |
|
|  |  |
| Новосибирск | | |
| 2024 | | |

**Исходные таблицы**

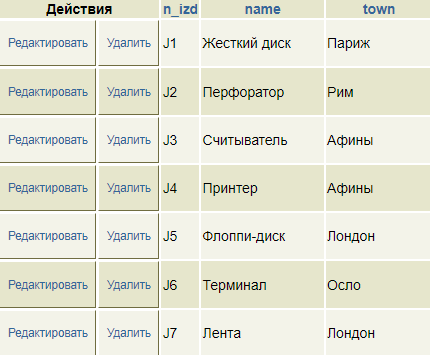
**Таблица s**

****

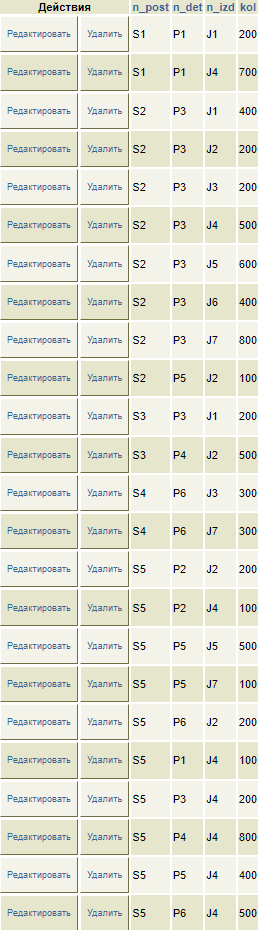
**Таблица p**

****

**Таблица j**

****

**Таблица spj**

****

**SQL Запросы**

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

SELECT COUNT (distinct spj.n\_izd) as count

FROM spj

WHERE spj.n\_post in (SELECT s.n\_post

FROM s

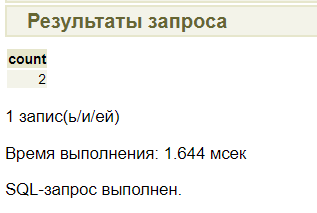
ORDER BY s.name

limit 1)

AND spj.n\_det in (SELECT p.n\_det

FROM p

WHERE p.ves>12)



1. 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 max(s.name) FROM s)

THEN (SELECT min(s.name) FROM s)

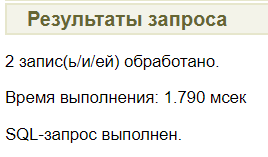
ELSE (SELECT max(s.name) FROM s)

END)

WHERE s.name = (SELECT max(s.name) FROM s)

OR s.name = (SELECT min(s.name) FROM s)



1. 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

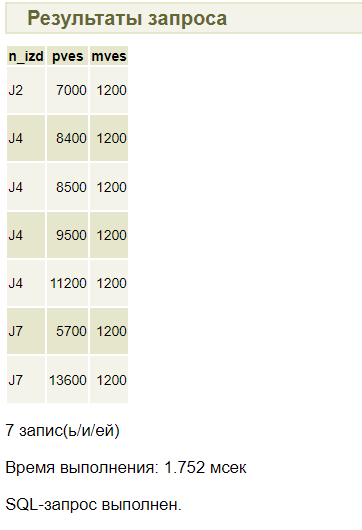
FROM spj t

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 \* b.mves

ORDER BY 1,2



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

SELECT DISTINCT s.n\_post

FROM s

EXCEPT

SELECT DISTINCT spj.n\_post

FROM spj

WHERE spj.n\_det IN (SELECT n\_det

FROM p

WHERE ves IN (SELECT MIN(ves) FROM p))



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

SELECT DISTINCT spj.n\_post, s.name, s.reiting, s.town

FROM spj

JOIN p ON p.n\_det = spj.n\_det

JOIN s ON s.n\_post = spj.n\_post

JOIN j ON j.n\_izd = spj.n\_izd

WHERE length(trim(j.name))>=7

AND

p.cvet='Красный'

EXCEPT

SELECT DISTINCT spj.n\_post, s.name, s.reiting, s.town

FROM spj

JOIN p ON p.n\_det = spj.n\_det

JOIN s ON s.n\_post = spj.n\_post

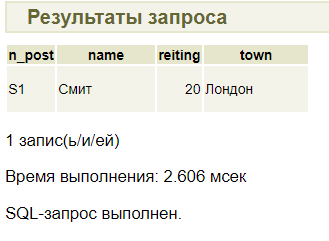
JOIN j ON j.n\_izd = spj.n\_izd

WHERE length(trim(j.name))<7

OR

p.cvet!='Красный'

ORDER BY 1



**Листинг программы**

**laba3.ec**

#include <stdio.h>

#include <stdlib.h>

// Вывод ошибок

void ErrorOutput(char \*st\_name, int errnum)

{

fprintf(stderr, "Ошибка %d в '%s'\n", errnum, st\_name);

fprintf(stderr, "Пояснение: %s\n", sqlca.sqlerrm.sqlerrmc);

char errorMessage[300];

if (sqlca.sqlerrd[1] != 0)

{

fprintf(stderr, "ISAM код: %d\n", sqlca.sqlerrd[1]);

rgetmsg(sqlca.sqlerrd[1], errorMessage, sizeof(errorMessage));

fprintf(stderr, "Сообщение ISAM: %s\n", errorMessage);

}

}

// Подключение к БД

void DatabaseConnect()

{

EXEC SQL CONNECT TO students@fpm2.ami.nstu.ru USER "pmi-b0706" USING "fodWiOc6";

// Обработка ошибки подключения к БД

if (sqlca.sqlcode != 0)

{

ErrorOutput("подключение к БД", sqlca.sqlcode);

exit(EXIT\_FAILURE);

}

fprintf(stdout, "\nПодключение к БД установлено\n");

}

// Подключение к схеме

void SchemaConnect()

{

EXEC SQL SET SEARCH\_PATH TO pmib0706;

// Обработка ошибки подключения к схеме

if (sqlca.sqlcode != 0)

{

ErrorOutput("подключение к схеме БД", sqlca.sqlcode);

exit(EXIT\_FAILURE);

}

fprintf(stdout, "Подключение к схеме установлено\n");

return;

}

// Выполнение первого запроса

void FirstTask()

{

fprintf(stdout, "\n\nПервый запрос:\n");

// Глобальная переменная

EXEC SQL BEGIN DECLARE SECTION;

int count;

EXEC SQL END DECLARE SECTION;

fprintf(stdout, "Выдать число изделий, для которых детали с весом больше 12 поставлял первый по алфавиту поставщик.\n");

// Запуск транзакции

EXEC SQL BEGIN WORK;

// SQL запрос

EXEC SQL SELECT COUNT(DISTINCT spj.n\_izd)

INTO :count

FROM spj

WHERE spj.n\_post in (SELECT s.n\_post

FROM s

order by s.name

limit 1)

AND spj.n\_det in (SELECT p.n\_det

FROM p

WHERE p.ves>12);

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

ErrorOutput("запрос на SELECT", sqlca.sqlcode);

// Отмена транзакции

EXEC SQL ROLLBACK WORK;

fprintf(stderr, "\nОтмена транзакции! Ошибка!\n");

}

else

{

fprintf(stdout, "\nЗапрос выполнен!\n");

fprintf(stdout, "count: %d\n", count);

// Подтверждение транзакции

EXEC SQL COMMIT WORK;

}

return;

}

// Выполнение второго запроса

void SecondTask()

{

fprintf(stdout, "\n\nВторой запрос:\n");

fprintf(stdout, "Поменять местами фамилии первого и последнего по алфавиту поставщика, т. е. первому по алфавиту поставщику установить фамилию последнего по алфавиту поставщика и наоборот.\n");

// Запуск транзакции

EXEC SQL BEGIN WORK;

// SQL запрос

EXEC SQL

UPDATE s SET name = (CASE WHEN s.name = (SELECT max(s.name) FROM s)

THEN (SELECT min(s.name) FROM s)

ELSE (SELECT max(s.name) FROM s)

END)

WHERE s.name = (SELECT max(s.name) FROM s)

OR s.name = (SELECT min(s.name) FROM s);

fprintf(stdout, "\nЗапрос выполнен ");

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

ErrorOutput("запрос на UPDATE", sqlca.sqlcode);

// Отмена транзакции

EXEC SQL ROLLBACK WORK;

fprintf(stderr, "\nОтмена транзакции! Ошибка!\n");

}

else

{

fprintf(stdout, "Запрос выполнен!\n");

fprintf(stdout, "\nИзменено: %d записи\n", sqlca.sqlerrd[2]);

// Подтверждение транзакции

EXEC SQL COMMIT WORK;

}

return;

}

// Выполнение третьего запроса

void ThirdTask()

{

fprintf(stdout, "\n\nТретий запрос:\n");

// Глобальная переменная

EXEC SQL BEGIN DECLARE SECTION;

struct

{

char n\_izd[6];

int pves;

int mves;

} data;

EXEC SQL END DECLARE SECTION;

fprintf(stdout, "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.\n");

// Инициализация курсора

EXEC SQL DECLARE third CURSOR FOR SELECT a.n\_izd, a.kol\*pa.ves pves, b.mves

INTO :data.n\_izd, :data.pves, :data.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 \* b.mves

ORDER BY 1,2;

if(sqlca.sqlcode < 0)

{

// Обработка ошибки

ErrorOutput("Инициализация курсора", sqlca.sqlcode);

return;

}

// Запуск транзакции и открытие курсора

EXEC SQL BEGIN WORK;

EXEC SQL OPEN third;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

ErrorOutput("открытие курсора", sqlca.sqlcode);

// Закрытие курсора и отмена транзакции

EXEC SQL CLOSE third;

EXEC SQL ROLLBACK WORK;

fprintf(stderr, "\nОтмена транзакции! Ошибка!\n");

return;

}

fprintf(stdout, "\nКурсор открыт!\n");

fprintf(stdout, "Обработка запроса\n\n");

// Чтение курсора

EXEC SQL FETCH third;

// Когда позиция курсора находится после последней строки

if (sqlca.sqlcode == 100)

{

fprintf(stdout, "Данных не найдено\n");

}

else

{

int row = 0;

fprintf(stdout, "Изделие\tВес поставки\tМинимальный вес\n");

while (true)

{

fprintf(stdout, "%s\t\t%d\t\t%d\n", data.n\_izd, data.pves, data.mves);

row++;

// Чтение курсора

EXEC SQL FETCH third;

if (sqlca.sqlcode == 100)

{

break;

}

}

fprintf(stdout, "\n%d записей\n", row);

}

// Закрытие курсора и завершение транзакции

EXEC SQL CLOSE third;

EXEC SQL COMMIT WORK;

return;

}

// Выполнение четвертого запроса

void FourthTask()

{

fprintf(stdout, "\n\nЧетвертый запрос:\n");

// Глобальные переменные

EXEC SQL BEGIN DECLARE SECTION;

char n\_post[6];

EXEC SQL END DECLARE SECTION;

fprintf(stdout, "Выбрать поставщиков, не поставивших ни одной из деталей, имеющих наименьший вес.\n");

// Инициализация курсора

EXEC SQL DECLARE fourth CURSOR FOR SELECT DISTINCT spj.n\_post

INTO :n\_post

FROM spj

EXCEPT

SELECT DISTINCT spj.n\_post

FROM spj

WHERE spj.n\_det IN (SELECT n\_det

FROM p

WHERE ves IN (SELECT MIN(ves) FROM p));

if(sqlca.sqlcode < 0)

{

// Обработка ошибки

ErrorOutput("Инициализация курсора", sqlca.sqlcode);

return;

}

// Запуск транзакции и открытие курсора

EXEC SQL BEGIN WORK;

EXEC SQL OPEN fourth;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

ErrorOutput("открытие курсора", sqlca.sqlcode);

// Закрытие курсора и отмена транзакции

EXEC SQL CLOSE fourth;

EXEC SQL ROLLBACK work;

fprintf(stderr, "\nОтмена транзакции! Ошибка!\n");

return;

}

fprintf(stdout, "Курсор открыт!\n");

fprintf(stdout, "Обработка запроса\n\n");

// Чтение курсора

EXEC SQL FETCH fourth;

// Когда позиция курсора находится после последней строки

if (sqlca.sqlcode == 100)

{

fprintf(stdout, "Данных не найдено\n");

}

else

{

int row = 0;

fprintf(stdout, "Изделие\n");

while (true)

{

fprintf(stdout, "%s\n", n\_post);

row++;

// Чтение курсора

EXEC SQL FETCH fourth;

if (sqlca.sqlcode == 100)

{

break;

}

}

fprintf(stdout, "\n%d записей\n", row);

}

// Закрытие курсора и завершение транзакции

EXEC SQL CLOSE fourth;

EXEC SQL COMMIT WORK;

return;

}

// Выполнение пятого запроса

void FifthTask()

{

fprintf(stdout, "\n\nПятый запрос:\n");

// Глобальная переменная

EXEC SQL BEGIN DECLARE SECTION;

struct

{

char n\_post[2 \* 6 + 1];

char name[2 \* 20 + 1];

int reiting;

char town[2 \* 20 + 1];

} data;

EXEC SQL END DECLARE SECTION;

fprintf(stdout, "Выдать полную информацию о поставщиках, поставляющих ТОЛЬКО красные детали и только для изделия с длиной названия не меньше 7\n");

/\* Запрос \*/

EXEC SQL DECLARE fifth CURSOR FOR SELECT DISTINCT spj.n\_post, s.name, s.reiting, s.town

INTO :data.n\_post, :data.name, :data.reiting, :data.town

FROM spj

JOIN p ON p.n\_det = spj.n\_det

JOIN s ON s.n\_post = spj.n\_post

JOIN j ON j.n\_izd = spj.n\_izd

WHERE length(trim(j.name))>=7

AND

p.cvet='Красный'

EXCEPT

SELECT DISTINCT spj.n\_post, s.name, s.reiting, s.town

FROM spj

JOIN p ON p.n\_det = spj.n\_det

JOIN s ON s.n\_post = spj.n\_post

JOIN j ON j.n\_izd = spj.n\_izd

WHERE length(trim(j.name))<7

OR

p.cvet!='Красный'

ORDER BY 1;

if(sqlca.sqlcode < 0)

{

// Обработка ошибки

ErrorOutput("Инициализация курсора", sqlca.sqlcode);

return;

}

// Запуск транзакции и открытие курсора

EXEC SQL BEGIN WORK;

EXEC SQL OPEN fifth;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

ErrorOutput("открытие курсора", sqlca.sqlcode);

// Закрытие курсора и отмена транзакции

EXEC SQL CLOSE fifth;

EXEC SQL ROLLBACK WORK;

fprintf(stderr, "\nОтмена транзакции! Ошибка!\n");

return;

}

fprintf(stdout, "Курсор открыт!\n");

fprintf(stdout, "Обработка запроса\n\n");

EXEC SQL FETCH fifth;

// Когда позиция курсора находится после последней строки

if (sqlca.sqlcode == 100)

{

fprintf(stdout, "Данных не найдено\n");

}

else

{

int row = 0;

fprintf(stdout, "Номер\tИмя\t\tРейтинг\t\tГород\n");

while (true)

{

fprintf(stdout, "%s\t%s\t%d\t\t%s\n", data.n\_post, data.name, data.reiting, data.town);

row++;

EXEC SQL FETCH fifth;

if (sqlca.sqlcode == 100)

{

break;

}

}

fprintf(stdout, "\n%d записей\n", row);

}

// Закрытие курсора и завершение транзакции

EXEC SQL CLOSE fifth;

EXEC SQL COMMIT WORK;

return;

}

// главная функция

int main()

{

// Подключение к БД

DatabaseConnect();

SchemaConnect();

// номер запроса

int task;

while (true)

{

fprintf(stdout, "\n\nВыберите номер запроса {1-5} или 0 для выхода.\n");

scanf("%d", &task);

switch(task)

{

case 0:

return 0;

case 1:

FirstTask();

break;

case 2:

SecondTask();

break;

case 3:

ThirdTask();

break;

case 4:

FourthTask();

break;

case 5:

FifthTask();

break;

default:

break;

}

}

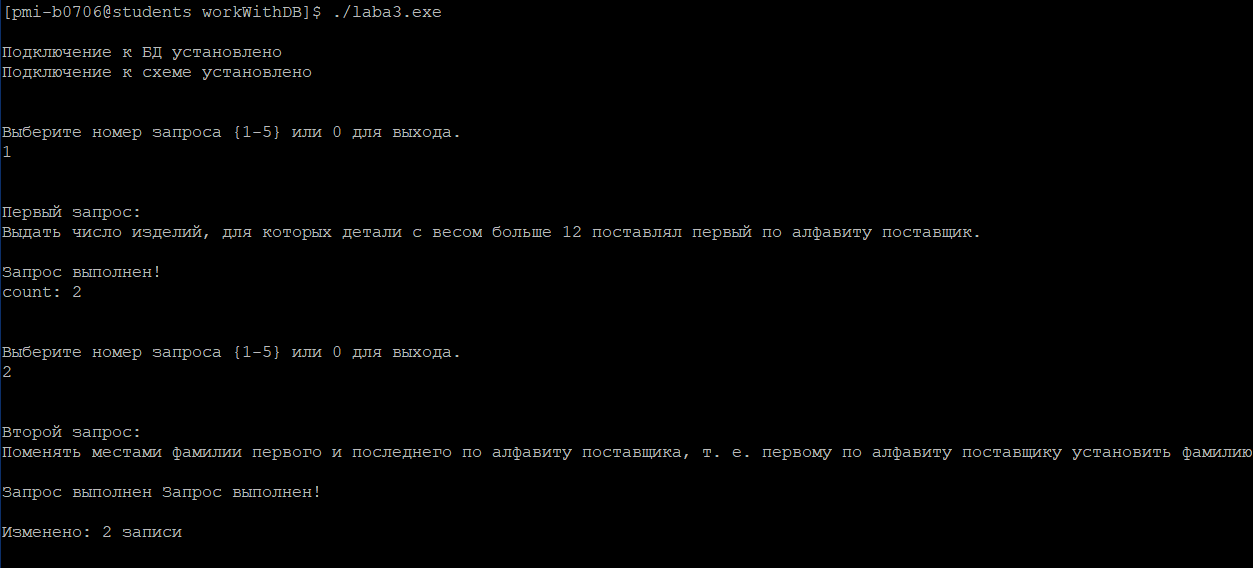
// Разъединение с источником данных

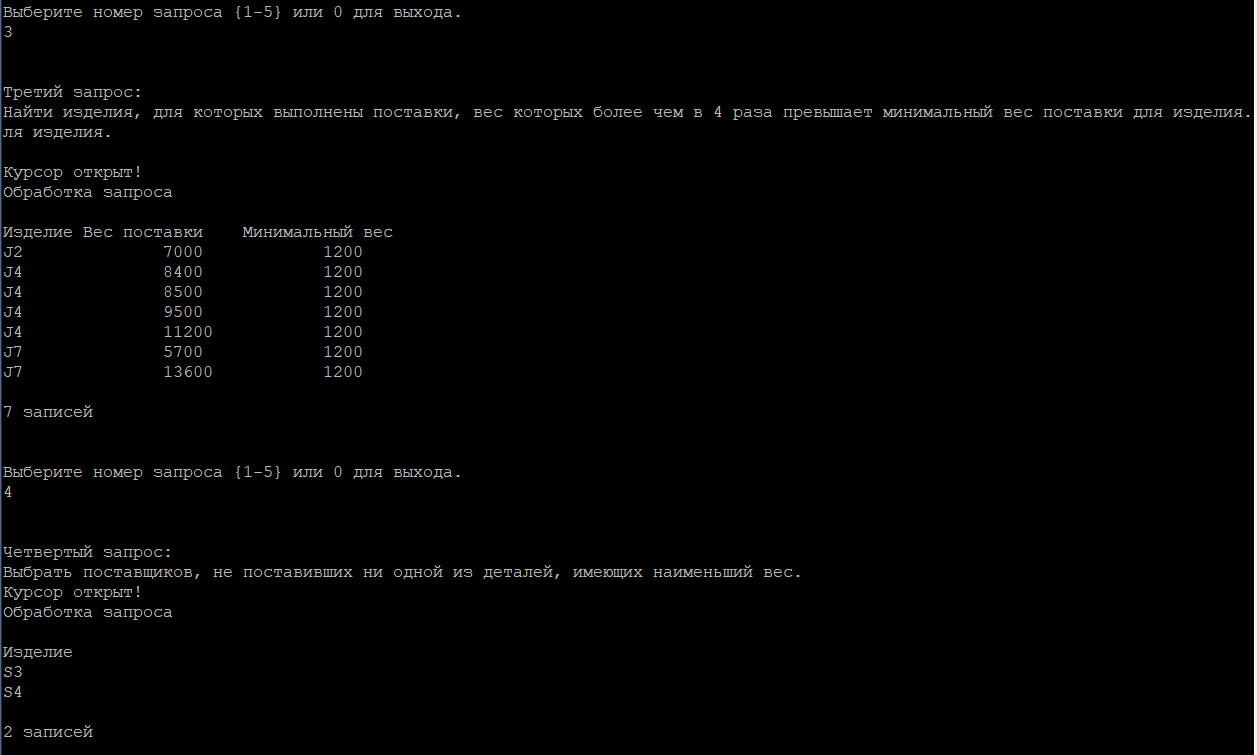
EXEC SQL DISCONNECT ALL;

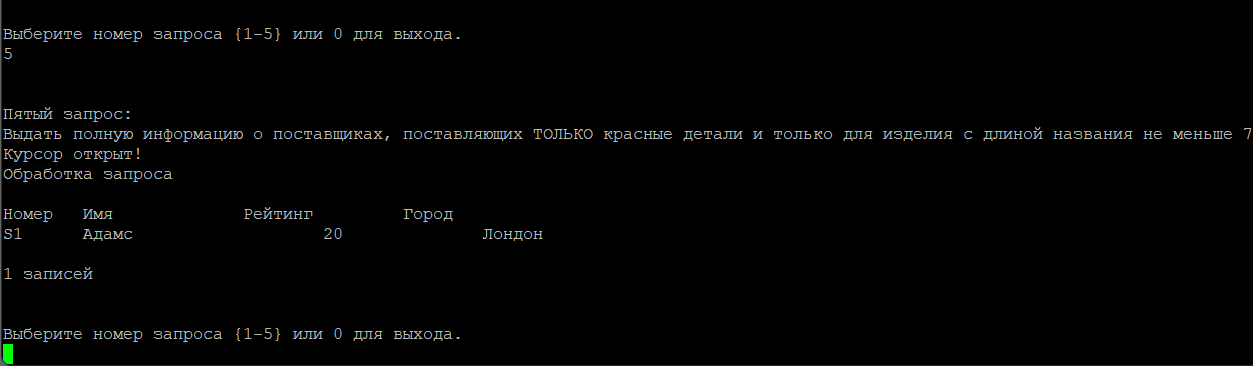
return 0;

}

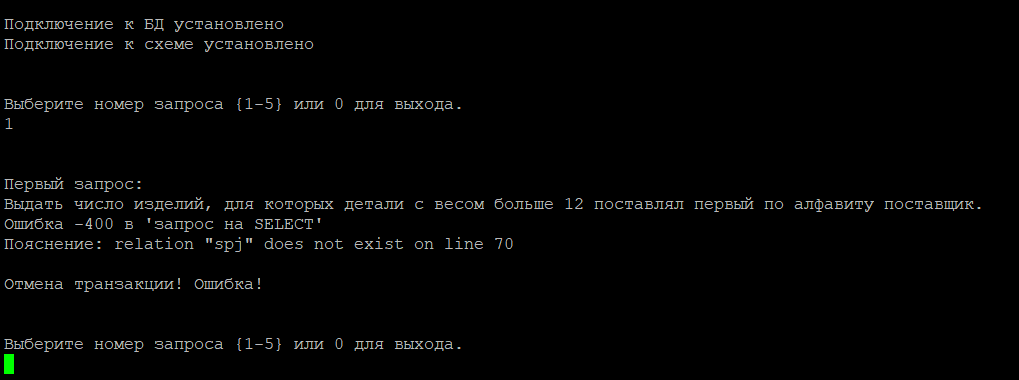
**Тестирование программы**

****

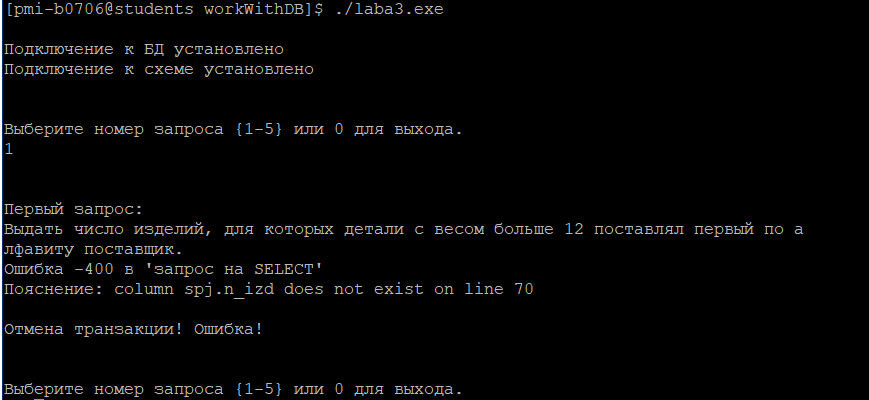
****

****

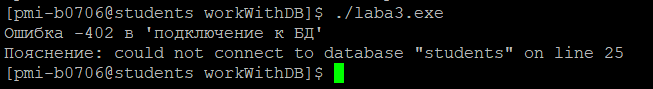
Если удалить таблицу spj:



Если таблица spj имеет другие столбцы:

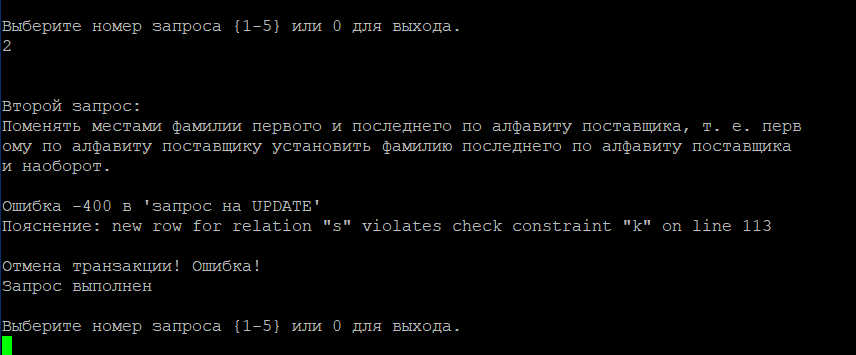


Неверный пароль:



Добавили ограничение на таблицу s:

ALTER TABLE s ADD CONSTRAINT k CHECK (n\_post != 'S1' OR name != 'Адамс');

  
Удалим содержание таблицы spj:

