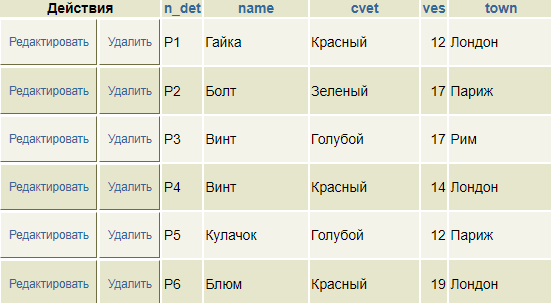
|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
|  | | |
| Лабораторная работа № 4 | | |
| 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 | | |
| 2023 | | |

**Исходные таблицы**

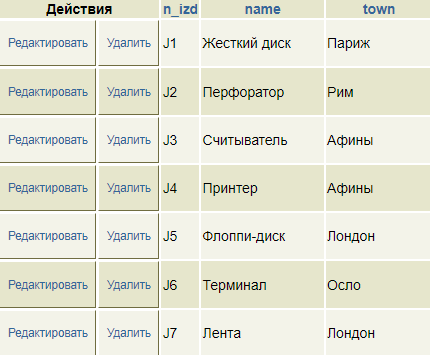
**Таблица s**

****

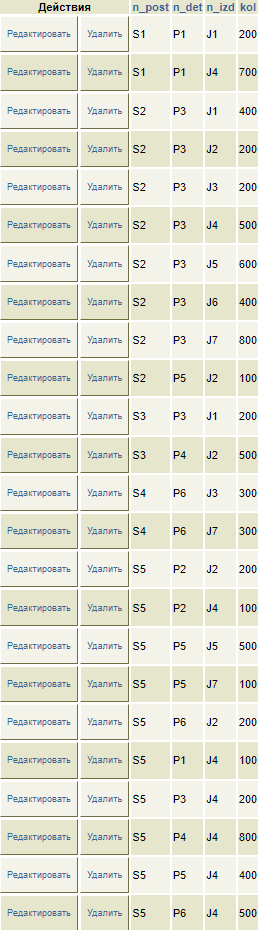
**Таблица p**

****

**Таблица j**

****

**Таблица spj**

****

**SQL Запросы**

1. Получить минимальный вес поставки для каждого изделия и найти их сумму.

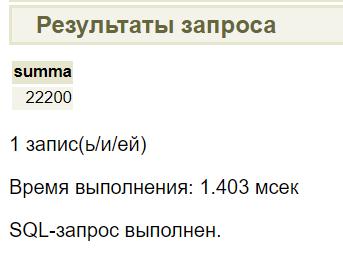
SELECT SUM(b.min\_ves) summa

FROM (SELECT MIN(t.kol\*p.ves) min\_ves

FROM spj t

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

GROUP BY t.n\_izd) b;



1. Для указанного поставщика S\* найти число поставок каждой детали, им поставлявшейся. Вывести номер детали, город детали, название детали, число поставок детали.

Для выполнения запроса в контексте статического SQL мы возьмем любого поставщика, например, S2:

SELECT a.n\_det, p.town, p.name, a.p\_sr

FROM (SELECT t.n\_det, COUNT(\*) p\_sr

FROM spj t

WHERE t.n\_post='S2'

GROUP BY t.n\_det) a

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

ORDER BY a.n\_det;



1. Ввести номер изделия J\*. Найти поставщиков, поставлявших детали для изделия J\*, и определить, какой процент составляет объем поставок каждого поставщика от общего количества деталей, поставленных для изделия. Вывести номер поставщика, объем поставок этого поставщика, общий объем поставок для изделия J\*, процент.

Для выполнения запроса в контексте статического SQL мы возьмем любое изделие, например, J1:

SELECT a.n\_post, a.p\_kol, b.o\_kol,

ROUND(a.p\_kol\*100.0/b.o\_kol,2) proc

FROM (SELECT t.n\_post, SUM(t.kol) p\_kol

FROM spj t

WHERE t.n\_izd= 'J1'

GROUP BY t.n\_post) a

CROSS JOIN (SELECT SUM(t.kol) o\_kol

FROM spj t

WHERE t.n\_izd= 'J1') b



**Листинг программы**

**laba3.ec**

#include <stdio.h>

#include <stdlib.h>

// Подключение к БД

int DatabaseConnect()

{

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

// Обработка ошибки подключения к БД

if (sqlca.sqlcode != 0)

{

fprintf(stderr, "Ошибка %d в подключении к БД\n", sqlca.sqlcode);

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

return 1;

}

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

return 0;

}

// Подключение к схеме

int SchemaConnect()

{

EXEC SQL SET SEARCH\_PATH TO pmib0706;

// Обработка ошибки подключения к схеме

if (sqlca.sqlcode != 0)

{

fprintf(stderr, "Ошибка %d в подключении к схеме БД\n", sqlca.sqlcode);

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

return 1;

}

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

return 0;

}

int main()

{

// Подключение к БД и схеме

if (DatabaseConnect() == 1)

{

exit(EXIT\_FAILURE);

}

if (SchemaConnect() == 1)

{

EXEC SQL DISCONNECT ALL;

exit(EXIT\_FAILURE);

}

// Глобальные переменные

EXEC SQL BEGIN DECLARE SECTION;

// Для первого запроса

int summa;

char task1[500] = "SELECT SUM(b.min\_ves) summa FROM (SELECT MIN(t.kol\*p.ves) min\_ves FROM spj t JOIN p ON p.n\_det=t.n\_det GROUP BY t.n\_izd) b";

// Для второго запроса

struct

{

char n\_det[14];

char town[40];

char name[40];

int p\_sr;

} data;

char n\_post[7];

char task2[500] = "SELECT a.n\_det, p.town, p.name, a.p\_sr FROM (SELECT t.n\_det, COUNT(\*) p\_sr FROM spj t WHERE t.n\_post= ? GROUP BY t.n\_det) a JOIN p ON a.n\_det = p.n\_det ORDER BY a.n\_det";

// Для третьего запроса

struct

{

char n\_post[7];

int p\_kol;

int o\_kol;

double proc;

} data2;

char n\_izd[6];

char task3[500] = "SELECT a.n\_post, a.p\_kol, b.o\_kol, ROUND(a.p\_kol\*100.0/b.o\_kol,2) proc FROM (SELECT t.n\_post, SUM(t.kol) p\_kol FROM spj t WHERE t.n\_izd= ? GROUP BY t.n\_post) a CROSS JOIN (SELECT SUM(t.kol) o\_kol FROM spj t WHERE t.n\_izd= ?) b";

EXEC SQL END DECLARE SECTION;

bool check1 = true, check2 = true, check3 = true;

EXEC SQL COMMIT WORK;

EXEC SQL BEGIN WORK;

// Подготовка первого SQL-запроса

EXEC SQL PREPARE task1 FROM :task1;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

fprintf(stderr, "Ошибка %d в подготовке первого запроса\n", sqlca.sqlcode);

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

EXEC SQL ROLLBACK WORK;

check1 = false;

}

else EXEC SQL COMMIT WORK;

EXEC SQL BEGIN WORK;

// Подготовка второго SQL-запроса

EXEC SQL PREPARE task2 FROM :task2;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

fprintf(stderr, "Ошибка %d в подготовке второго запроса\n", sqlca.sqlcode);

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

EXEC SQL ROLLBACK WORK;

check2 = false;

}

else EXEC SQL COMMIT WORK;

EXEC SQL BEGIN WORK;

// Подготовка SQL запроса

EXEC SQL PREPARE task3 FROM :task3;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

fprintf(stderr, "Ошибка %d в подготовке запроса\n", sqlca.sqlcode);

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

EXEC SQL ROLLBACK WORK;

check3 = false;

}

else EXEC SQL COMMIT WORK;

if (!check1 && !check2 && !check3)

{

fprintf(stderr, "Ни один из запросов не прошел подготовку\n");

EXEC SQL DISCONNECT ALL;

exit(EXIT\_FAILURE);

}

// Номер запроса

int number;

while (true)

{

fprintf(stdout, "\n\nВыберите номер запроса {1-3} или 0 для выхода.\n");

scanf("%d", &number);

switch(number)

{

case 0:

return 0;

case 1:

if(!check1)

{

fprintf(stdout, "Данный запрос не прошел подготовку, выберите другой запрос\n");

break;

}

// Выполнение первого запроса

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

fprintf(stdout, "Получить минимальный вес поставки для каждого изделия и найти их сумму.\n");

EXEC SQL BEGIN WORK;

// Выполнение SQL запроса

EXEC SQL EXECUTE task1 INTO :summa;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки и отмена транзакции

fprintf(stderr, "Ошибка %d в выполнении запроса\n", sqlca.sqlcode);

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

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

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

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

EXEC SQL COMMIT WORK;

break;

case 2:

if(!check2)

{

fprintf(stdout, "Данный запрос не прошел подготовку, выберите другой запрос\n");

break;

}

// Выполнение второго запроса

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

fprintf(stdout, "Для указанного поставщика S\* найти число поставок каждой детали, им поставлявшейся. Вывести номер детали, город детали, название детали, число поставок детали.\n");

int number1;

printf("\nВведите номер поставщика: ");

if(scanf("%d", &number1) != 1)

{

printf("Ошибка: введено не число\n");

break;

}

else

{

sprintf(n\_post, "S%d", number1);

}

EXEC SQL BEGIN WORK;

// Инициализация курсора

EXEC SQL DECLARE secondCursor CURSOR FOR task2;

if(sqlca.sqlcode < 0)

{

// Обработка ошибки и отмена транзакции

fprintf(stderr, "Ошибка %d в инициализации курсора\n", sqlca.sqlcode);

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

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

// Открытие курсора

EXEC SQL OPEN secondCursor USING :n\_post;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

fprintf(stderr, "Ошибка %d в открытии курсора\n", sqlca.sqlcode);

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

// Закрытие курсора

EXEC SQL CLOSE secondCursor;

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

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

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

// Чтение курсора

EXEC SQL FETCH secondCursor INTO :data.n\_det, :data.town, :data.name, :data.p\_sr;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

fprintf(stderr, "Ошибка %d в чтении курсора\n", sqlca.sqlcode);

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

// Закрытие курсора

EXEC SQL CLOSE secondCursor;

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

if (sqlca.sqlcode == 100)

{

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

}

else

{

int row = 0;

printf("%-30s%-25s%-27s%s\n", "Номер детали", "Город", "Название", "Число поставок");

while (true)

{

fprintf(stdout, "%-19s%-5s%-10s%d\n", data.n\_det, data.town, data.name, data.p\_sr);

row++;

// Чтение курсора

EXEC SQL FETCH secondCursor INTO :data.n\_det, :data.town, :data.name, :data.p\_sr;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

fprintf(stderr, "Ошибка %d в чтении курсора\n", sqlca.sqlcode);

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

// Закрытие курсора

EXEC SQL CLOSE secondCursor;

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

if (sqlca.sqlcode == 100)

{

break;

}

}

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

}

// Закрытие курсора

EXEC SQL CLOSE secondCursor;

EXEC SQL COMMIT WORK;

break;

case 3:

if(!check3)

{

fprintf(stdout, "Данный запрос не прошел подготовку, выберите другой запрос\n");

break;

}

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

fprintf(stdout, "Ввести номер изделия J\*. Найти поставщиков, поставлявших детали для изделия J\*, и определить, какой процент составляет объем поставок каждого поставщика от общего количества деталей, поставленных для изделия. Вывести номер поставщика, объем поставок этого поставщика, общий объем поставок для изделия J\*, процент.\n");

int number2;

printf("\nВведите номер изделия: ");

if(scanf("%d", &number2) != 1)

{

printf("Ошибка: введено не число\n");

break;

}

else

{

sprintf(n\_izd, "J%d", number2);

}

EXEC SQL BEGIN WORK;

// Подготовка SQL запроса

EXEC SQL PREPARE task3 FROM :task3;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки и отмена транзакции

fprintf(stderr, "Ошибка %d в подготовке запроса\n", sqlca.sqlcode);

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

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

// Инициализация курсора

EXEC SQL DECLARE thirdCursor CURSOR FOR task3;

if(sqlca.sqlcode < 0)

{

// Обработка ошибки и отмена транзакции

fprintf(stderr, "Ошибка %d в инициализации курсора\n", sqlca.sqlcode);

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

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

// Открытие курсора

EXEC SQL OPEN thirdCursor USING :n\_izd, :n\_izd;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

fprintf(stderr, "Ошибка %d в открытии курсора\n", sqlca.sqlcode);

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

// Закрытие курсора

EXEC SQL CLOSE thirdCursor;

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

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

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

// Чтение курсора

EXEC SQL FETCH thirdCursor INTO :data2.n\_post, :data2.p\_kol, :data2.o\_kol, :data2.proc;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

fprintf(stderr, "Ошибка %d в чтении курсора\n", sqlca.sqlcode);

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

// Закрытие курсора

EXEC SQL CLOSE thirdCursor;

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

if (sqlca.sqlcode == 100)

{

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

}

else

{

int row = 0;

printf("%-35s%-35s%-30s%s\n", "Номер поставщика", "Объем поставок", "Общий объем", "Процент");

while (true)

{

printf("%-20s%-22d%-20d%f\n", data2.n\_post, data2.p\_kol, data2.o\_kol, data2.proc);

row++;

// Чтение курсора

EXEC SQL FETCH thirdCursor INTO :data2.n\_post, :data2.p\_kol, :data2.o\_kol, :data2.proc;

if (sqlca.sqlcode < 0)

{

// Обработка ошибки

fprintf(stderr, "Ошибка %d в чтении курсора\n", sqlca.sqlcode);

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

// Закрытие курсора

EXEC SQL CLOSE thirdCursor;

// Откат транзакции

EXEC SQL ROLLBACK WORK;

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

break;

}

if (sqlca.sqlcode == 100)

{

break;

}

}

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

}

// Закрытие курсора

EXEC SQL CLOSE thirdCursor;

EXEC SQL COMMIT WORK;

break;

default:

break;

}

}

// Разъединение с источником данных

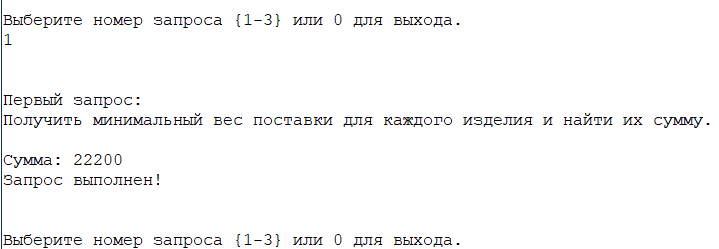
EXEC SQL DISCONNECT ALL;

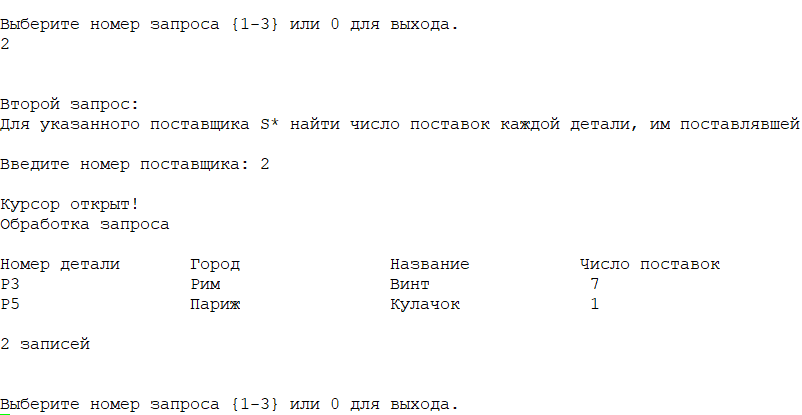
exit(0);

}

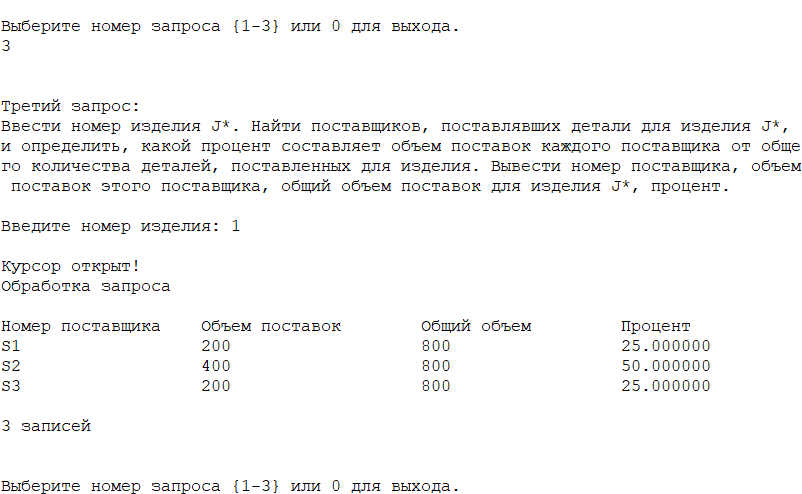
**Тестирование программы**

1 запрос:

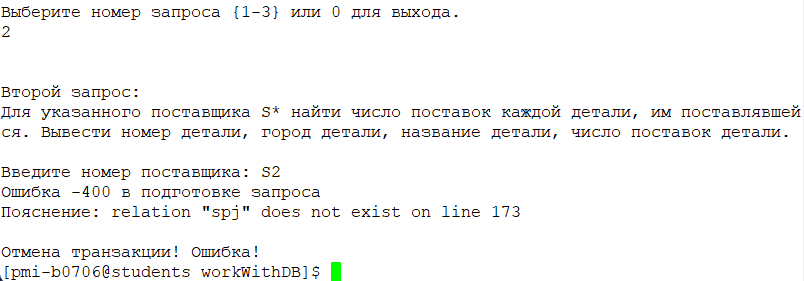
****

2 запрос:

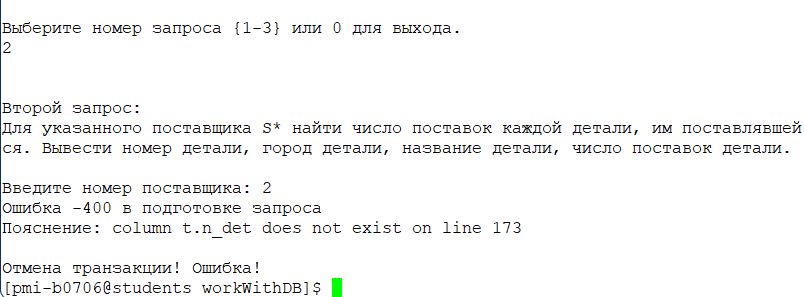
3 запрос:

****

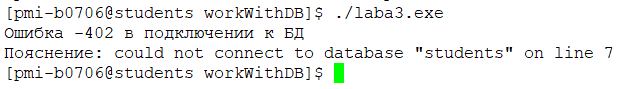
Работа программы, если удалить таблицу spj:

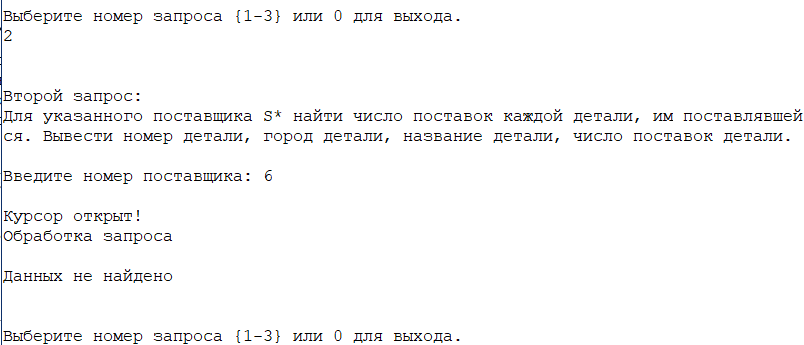


Работа программы, если таблица spj имеет другие столбцы:

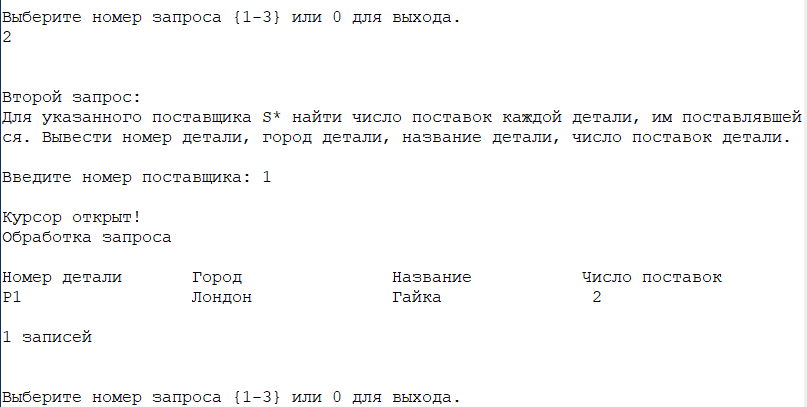


Работа программы при вводе неверного пароля:

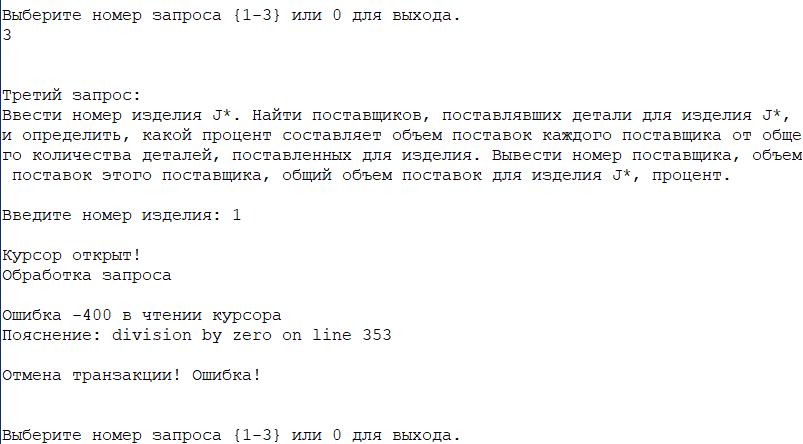
  
Работа курсора, если нет строк для вывода:



Работа курсора при выводе одной строки:

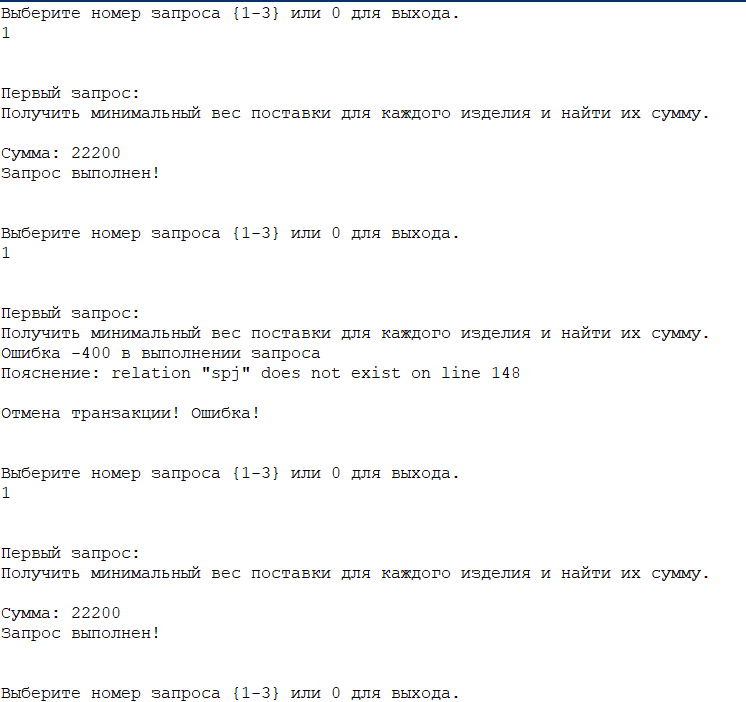


Проверка деления на ноль в запросе:



Проверка работы программы при порче данных в таблице во время выполнения:

1. Выполним 1 запрос
2. Удалим таблицу spj и выполним 1 запрос
3. Восстановим таблицу и выполним 1 запрос



Проверим работы программы, если 2 запроса из 3 не будут успешно подготовлены:

-Удалим таблицу p, чтобы первые 2 запроса не были успешно подготовлены

