|  |  |  |
| --- | --- | --- |
| 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» | | |
| «РАБОТА С БАЗОЙ ДАННЫХ СРЕДСТВАМИ ВСТРОЕННОГО SQL» | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-01 |
| Бригада: | 8 |
| Вариант: | 3 |
| Студенты: | Куликов Данил |
|  | Иовенко Дмитрий |
| Преподаватели: | Стасышина Т.Л. |
|  | Сивак М.А. |
|  |
|
| Новосибирск | | |
| 2023 | | |

1. **Цель работы**

Приобрести практические навыки работы со встроенным SQL средствами ESQL/C, включая использование курсоров и средств динамического SQL.

1. **Содержание работы**

**Выполнение запросов на исходных данных:**

* 1. Выдать число деталей, поставлявшихся для изделий, у которых есть поставки с весом от 5000 до 6000.

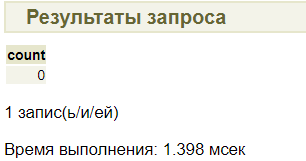
Запрос:

SELECT COUNT(DISTINCT spj.n\_det)

FROM spj

WHERE spj.kol BETWEEN 5000 AND 6000;

Результат выполнения запроса:



* 1. Поменять местами вес деталей из Рима и из Парижа, т. е. деталям из Рима установить вес детали из Парижа, а деталям из Парижа установить вес детали из Рима. Если деталей несколько, брать наименьший вес.

Запрос:

UPDATE p

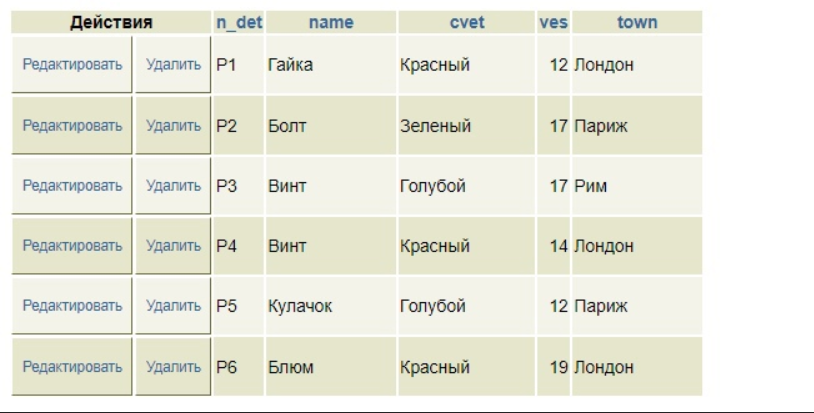
SET ves = (SELECT MIN(ves)

FROM p AS p2

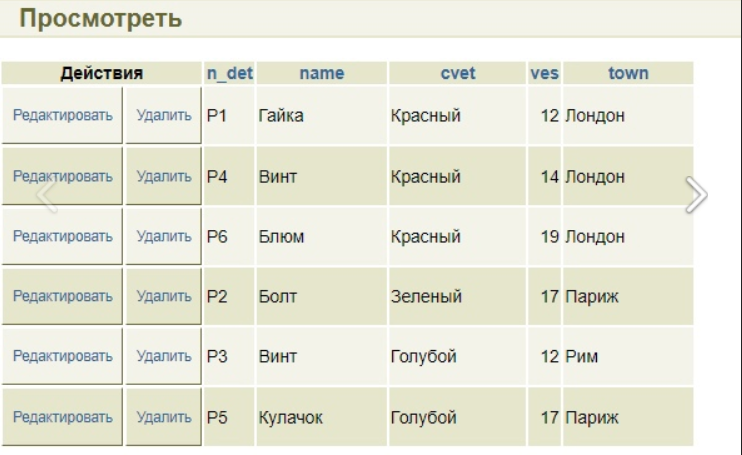
WHERE (p2.town = 'Париж' AND p.town = 'Рим') OR (p2.town = 'Рим' AND p.town = 'Париж'))

WHERE town IN ('Рим', 'Париж');

Начальные данные:



Результат выполнения запроса:



* 1. Найти детали, имеющие поставки, объем которых не превышает половину максимального объема поставки этой детали поставщиком из Парижа.  
       
     Запрос:

SELECT DISTINCT p1.n\_det

FROM p p1

JOIN spj ON p1.n\_det = spj.n\_det

WHERE 2\*spj.kol <= (SELECT MAX(spj.kol)

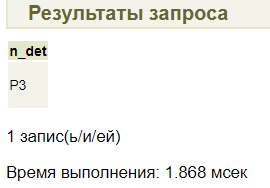
FROM p

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

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

WHERE s.town = 'Париж' AND p.n\_det = p1.n\_det);

Результат:



* 1. Выбрать поставщиков, не поставивших ни одной из деталей, поставляемых для изделий из Парижа.  
       
     Запрос:

SELECT DISTINCT s.n\_post, s.name

FROM s

WHERE s.n\_post NOT IN (SELECT DISTINCT spj.n\_post

FROM spj

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

WHERE j.town = 'Париж');

Результат:



* 1. Выдать полную информацию о деталях, которые поставлялись ТОЛЬКО поставщиками, проживающими в Афинах.

Запрос:   
SELECT p.\*

FROM p

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

FROM spj

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

WHERE s.town = 'Афины')

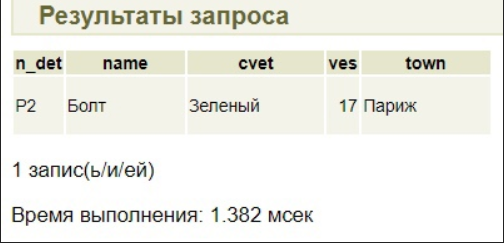
AND p.n\_det NOT IN (SELECT spj.n\_det

FROM spj

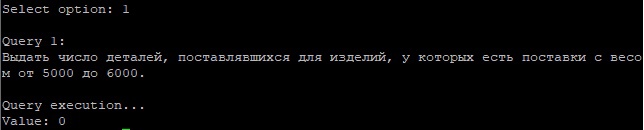
JOIN s ON spj.n\_post = s.n\_post

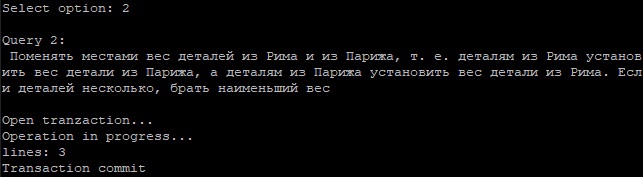
WHERE s.town <> 'Афины');

Результат:

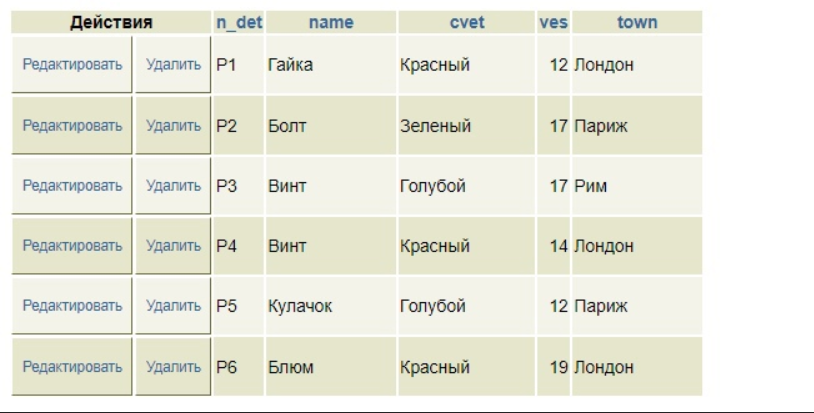


Результаты работы программы:

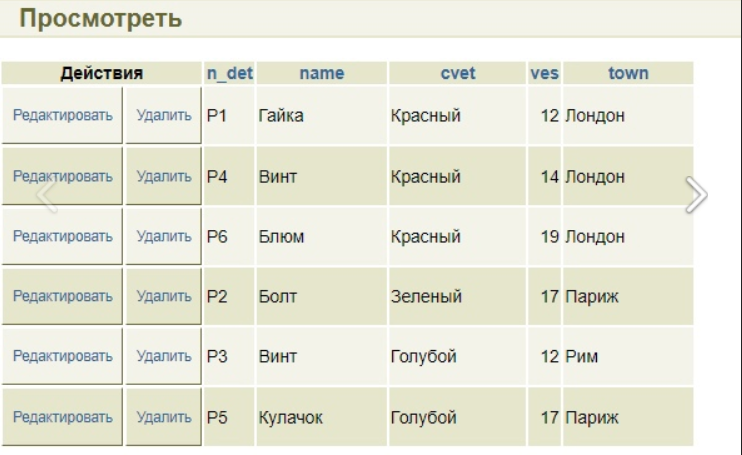
1. Выдать число деталей, поставлявшихся для изделий, у которых есть поставки с весом от 5000 до 6000.   
   
2. Поменять местами вес деталей из Рима и из Парижа, т. е. деталям из Рима установить вес детали из Парижа, а деталям из Парижа установить вес детали из Рима. Если деталей несколько, брать наименьший вес.

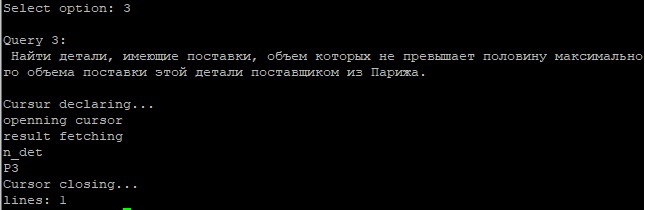
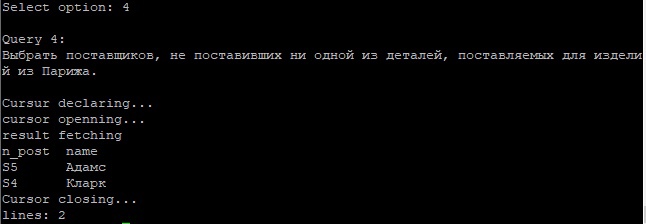
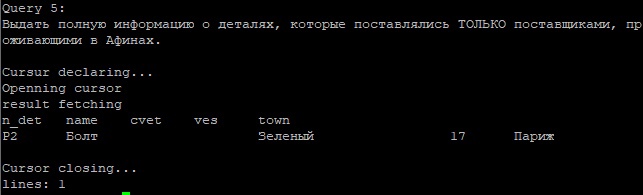


Исходные данные:



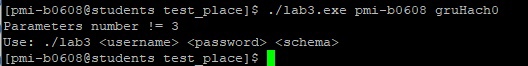
Результат:



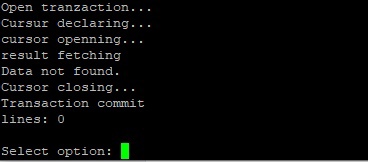
1. Найти детали, имеющие поставки, объем которых не превышает половину максимального объема поставки этой детали поставщиком из Парижа.  
   
2. Выбрать поставщиков, не поставивших ни одной из деталей, поставляемых для изделий из Парижа.  
   
3. Выдать полную информацию о деталях, которые поставлялись ТОЛЬКО поставщиками, проживающими в Афинах.  
   

1. **Тесты**

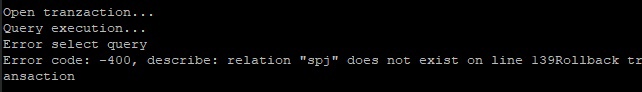
* Передача неверного количества аргументов



* Отсутствие подходящих данных



* Обращение к несуществующей таблице



1. **Код программы**

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <sqltypes.h>

#include <string.h>

#define AUTH\_EXCEPTION 1

#define DEFAULT\_USER "pmi-b0608"

#define DEFAULT\_PWD "gruHach0"

#define DEFAULT\_SCHEMA "pmib0608"

#define USER\_LEN 12

#define PWD\_LEN 12

#define SCHEMA\_LEN 11

// объявление главных переменных

EXEC SQL begin declare section;

char user\_name[USER\_LEN] = DEFAULT\_USER;

char password[PWD\_LEN] = DEFAULT\_PWD;

char schema[SCHEMA\_LEN] = DEFAULT\_SCHEMA;

unsigned int result;

char num [ 6 ] ;

char name [ 30 ] ;

int ves;

char cvet [30];

char town [ 30 ] ;

int kol;

EXEC SQL end declare section;

void display\_options() {

printf("\n\n\tAviable options\n");

printf("1) Execute query №1\n");

printf("2) Execute query №2\n");

printf("3) Execute query №3\n");

printf("4) Execute query №4\n");

printf("5) Execute query №5\n");

printf("6) Display options\n");

printf("0) Exit\n\n");

}

void close\_programm () {

EXEC SQL DISCONNECT CURRENT;

printf("Base disconnected\n");

exit(EXIT\_SUCCESS);

}

void handle\_error(char \*operation){

if (sqlca.sqlcode < 0) {

printf("Ошибка подключения к базе данных.\n");

exit(EXIT\_FAILURE);

}

else {

printf("Подключено к базе данных \"students\" под пользователем %s\n", user\_name);

}

}

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

{

printf("Error %s\n", st\_name);

printf("Error code: %d, describe: %s", errnum, sqlca.sqlerrm.sqlerrmc);

}

int main (int argc, const char\* argv) {

if( argc == 4){

strncpy(user\_name, argv[1], USER\_LEN-1);

strncpy(password, argv[2], PWD\_LEN-1);

strncpy(schema, argv[3], SCHEMA\_LEN-1);

}

EXEC SQL CONNECT TO students@fpm2.ami.nstu.ru USER :user\_name USING :password;

if (sqlca.sqlcode < 0) {

printf("DB connection error\n");

exit(1);

}

else {

printf("Connection to DB - successful under %s\n", user\_name);

}

EXEC SQL begin; // начать транзакцию

EXEC SQL SET search\_path TO :schema; // Выбор схемы

EXEC SQL commit; // закончить транзацию

int cmd = 1;

display\_options();

while (cmd != 0) {

printf("\nSelect option: ");

scanf("%d", &cmd);

switch(cmd) {

case 1:

task\_1();

break;

case 2:

task\_2();

break;

case 3:

task\_3();

break;

case 4:

task\_4();

break;

case 5:

task\_5();

break;

case 6:

display\_options();

break;

case 0:

close\_programm();

break;

default:

printf("\n");

}

}

return 0;

}

void task\_1() {

printf("\nQuery 1:\n");

printf("Выдать число деталей, поставлявшихся для изделий, у которых есть поставки с весом от 5000 до 6000. \n\n");

printf("Open tranzaction...\n");

EXEC SQL begin work;

printf("Query execution...\n");

EXEC SQL SELECT COUNT(DISTINCT spj.n\_det) into :result

FROM spj

WHERE spj.kol BETWEEN 5000 AND 6000;

if (sqlca.sqlcode < 0) {

log\_error("select query", sqlca.sqlcode);

printf("Rollback transaction\n");

EXEC SQL rollback work; // отменяем транзакцию

return;

}

printf("Value: %d \n", result);

printf("Transaction commit\n");

EXEC SQL commit work;

return;

}

void task\_2() {

printf("\nQuery 2:\n");

printf(" Поменять местами вес деталей из Рима и из Парижа, т. е. деталям из Рима установить вес детали из Парижа, а деталям из Парижа установить вес детали из Рима. Если деталей несколько, брать наименьший вес \n\n");

printf("Open tranzaction...\n");

EXEC SQL begin work;

printf("Operation in progress\n");

EXEC SQL UPDATE p

SET ves = (SELECT MIN(ves)

FROM p AS p2

WHERE (p2.town = 'Париж' AND p.town = 'Рим') OR (p2.town = 'Рим' AND p.town = 'Париж'))

WHERE town IN ('Рим', 'Париж');

if (sqlca.sqlcode < 0) {

log\_error("update query", sqlca.sqlcode);

printf("Rollback transaction\n");

EXEC SQL rollback work;

return;

}

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

printf("Transaction commit\n");

EXEC SQL commit work;

return;

}

void task\_3() {

printf("\nQuery 3:\n");

printf(" Найти детали, имеющие поставки, объем которых не превышает половину максимального объема поставки этой детали поставщиком из Парижа. \n\n");

printf("Open tranzaction...\n");

EXEC SQL begin work;

printf("Cursur declaring...\n");

EXEC SQL DECLARE cursor\_1 CURSOR FOR

SELECT DISTINCT p1.n\_det

FROM p p1

JOIN spj ON p1.n\_det = spj.n\_det

WHERE 2\*spj.kol <= (SELECT MAX(spj.kol)

FROM p

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

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

WHERE s.town = 'Париж' AND p.n\_det = p1.n\_det);

// Проверяем объявление курсора

if(sqlca.sqlcode < 0)

{

log\_error("Cursur declaring", sqlca.sqlcode);

printf("Rollback transaction\n");

EXEC SQL ROLLBACK WORK;

return;

}

else

{

printf("openning cursor\n");

EXEC SQL OPEN cursor\_1;

if (sqlca.sqlcode < 0){

log\_error("cursor openning", sqlca.sqlcode);

printf("Rollback transaction\n");

EXEC SQL rollback work;

}

else {

printf("result fetching\n");

EXEC SQL FETCH cursor\_1 INTO :num;

int row\_count=0;

if(sqlca.sqlcode < 0){

log\_error("fetching result", sqlca.sqlcode);

printf("Rollback transaction\n");

EXEC SQL rollback work;

return;

}

else if(sqlca.sqlcode == 100)

printf("Data not found.\n");

else

{

printf("n\_det\n");

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

row\_count++;

while(sqlca.sqlcode == 0){

EXEC SQL FETCH cursor\_1 INTO :num;

if (sqlca.sqlcode == 0) {

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

row\_count++;

}

};

};

printf("Cursor closing...\n");

EXEC SQL close cursor\_1;

printf("Transaction commit\n");

EXEC SQL commit work;

printf("lines: %d \n", row\_count);

return;

}

}

}

void task\_4() {

printf("\nQuery 4:\n");

printf("Выбрать поставщиков, не поставивших ни одной из деталей, поставляемых для изделий из Парижа.\n\n");

printf("Open tranzaction...\n");

EXEC SQL begin work;

printf("Cursur declaring...\n");

EXEC SQL DECLARE cursor\_2 CURSOR FOR

SELECT DISTINCT s.n\_post, s.name

FROM s

WHERE s.n\_post NOT IN (SELECT DISTINCT spj.n\_post

FROM spj

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

WHERE j.town = 'Париж');

// Проверяем объявление курсора

if(sqlca.sqlcode < 0)

{

log\_error("Cursur declaring", sqlca.sqlcode);

EXEC SQL ROLLBACK WORK;

return;

}

else

{

printf("cursor openning...\n");

EXEC SQL OPEN cursor\_2;

if (sqlca.sqlcode < 0){

log\_error("cursor openning", sqlca.sqlcode);

EXEC SQL rollback work;

}

else{

printf("result fetching\n");

EXEC SQL FETCH cursor\_2 INTO :num, :name; // извлекаем строку

int row\_count=0;

if(sqlca.sqlcode < 0){

log\_error("fetching result", sqlca.sqlcode);

EXEC SQL rollback work;

return;

}

else if(sqlca.sqlcode == 100)

printf("Data not found.\n");

else{

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

printf("%s\t%s\n",num,name);

row\_count++;

while(sqlca.sqlcode == 0){

EXEC SQL FETCH cursor\_2 INTO :num, :name;

if(sqlca.sqlcode == 0){

printf("%s\t%s\n",num, name);

row\_count++;

}

}

};

printf("Cursor closing...\n");

EXEC SQL close cursor\_2;

printf("Transaction commit\n");

EXEC SQL commit work;

printf("lines: %d \n", row\_count);

return;

}

}

}

void task\_5() {

printf("\nQuery 5:\n");

printf("Выдать полную информацию о деталях, которые поставлялись ТОЛЬКО поставщиками, проживающими в Афинах.\n\n");

printf("Open tranzaction...\n");

EXEC SQL begin work;

printf("Cursur declaring...\n");

EXEC SQL DECLARE cursor\_3 CURSOR FOR

SELECT p.\*

FROM p

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

FROM spj

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

WHERE s.town = 'Афины')

AND p.n\_det NOT IN (SELECT spj.n\_det

FROM spj

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

WHERE s.town <> 'Афины');

// Проверяем объявление курсора

if(sqlca.sqlcode < 0)

{

log\_error("Cursur declaring", sqlca.sqlcode);

EXEC SQL ROLLBACK WORK;

return;

}

else

{

printf("Openning cursor\n");

EXEC SQL OPEN cursor\_3;

if (sqlca.sqlcode < 0){

log\_error("cursor openning", sqlca.sqlcode);

EXEC SQL rollback work;

}

else{

printf("result fetching\n");

EXEC SQL FETCH cursor\_3 INTO :num, :name, :cvet, :ves, :town; // переводим курсор к определенной строке активного множества и выбираем значения из этой строки

int row\_count=0;

if(sqlca.sqlcode < 0){

log\_error("и).", sqlca.sqlcode);

EXEC SQL rollback work;

return;

}

else if(sqlca.sqlcode == 100)

printf("Data not found\n");

else{

printf("n\_det\tname\tcvet\tves\ttown\n");

printf("%s\t%s\t%s\t%d\t%s\n", num, name, cvet, ves, town);

row\_count++;

while(sqlca.sqlcode == 0){

EXEC SQL FETCH cursor\_3 INTO :num, :name, :cvet, :ves, :town;

if(sqlca.sqlcode == 0){

printf("%s\t%s\t%s\t%d\t%s\n", num, name, cvet, ves, town);

row\_count++;

}

}

};

printf("Cursor closing...\n");

EXEC SQL close cursor\_3;

printf("Transaction commit\n");

EXEC SQL commit work;

printf("lines: %d \n", row\_count);

return;

}

}

}