|  |  |  |
| --- | --- | --- |
| 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» | | |
| «Динамический 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 средствами динамического SQL при написании программ на ESQL/C.

1. **Содержание работы**

**Выполнение запросов на исходных данных:**

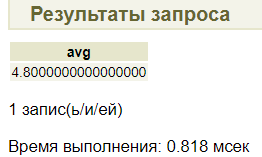
1. Получить число поставок для каждого поставщика и найти их среднее

**SELECT AVG(t.post\_count)**

**FROM (SELECT COUNT(\*) post\_count**

**FROM spj**

**GROUP BY spj.n\_post) t;**



2. Для каждого изделия из указанного города найти суммарный объем поставок по каждой детали, для него поставлявшейся. Вывести номер изделия, название изделия, номер детали, название детали, цвет детали, суммарный объем поставок детали для изделия.

**SELECT t.n\_izd, i.name, t.n\_det, d.name, d.cvet, t.kol\_sum**

**FROM (SELECT j.n\_izd, spj.n\_det, SUM(spj.kol) kol\_sum**

**FROM j**

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

**WHERE j.town = 'Афины'**

**GROUP BY j.n\_izd, spj.n\_det) t**

**JOIN j i ON i.n\_izd = t.n\_izd**

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

**ORDER BY t.n\_izd, t.n\_det**

****

3. Ввести номер детали P\*. Найти города, в которые поставлялась деталь P\*, и определить, какой процент составляют поставки в каждый город от общего числа поставок детали P\*. Вывести город, число поставок деталей в этот город, общее число поставок детали P\*, процент.

**SELECT J.town,**

**COUNT(\*) AS Kol\_Post, (SELECT COUNT(\*)**

**FROM SPJ**

**WHERE n\_det = 'P3') AS Obch\_kol\_post,**

**COUNT(\*) \* 100.0 / (SELECT COUNT(\*)**

**FROM SPJ**

**WHERE n\_det = 'P3') AS Procent**

**FROM J**

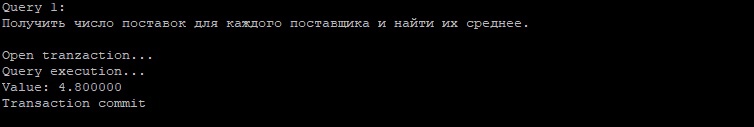
**JOIN SPJ ON J.n\_izd = SPJ.n\_izd**

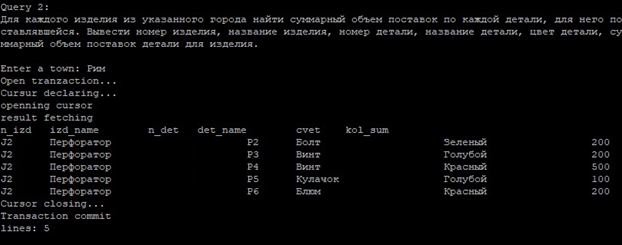
**WHERE SPJ.n\_det = 'P3'**

**GROUP BY J.town;**

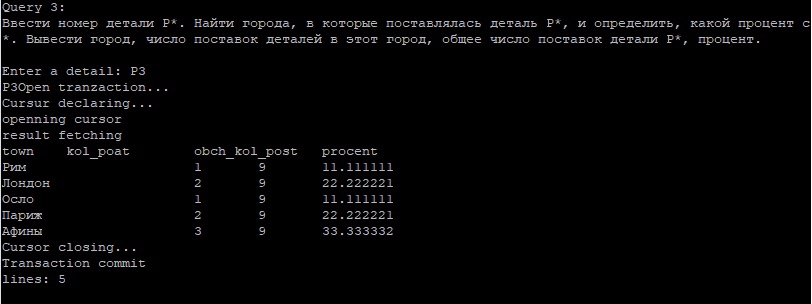
****

Результаты работы программы:

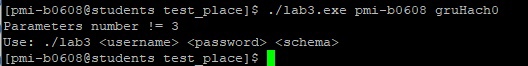
1. Получить число поставок для каждого поставщика и найти их среднее
2. Для каждого изделия из указанного города найти суммарный объем поставок по каждой детали, для него поставлявшейся. Вывести номер изделия, название изделия, номер детали, название детали, цвет детали, суммарный объем поставок детали для изделия.



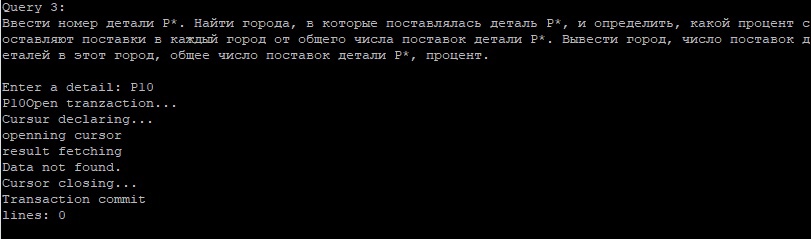
1. Ввести номер детали P\*. Найти города, в которые поставлялась деталь P\*, и определить, какой процент составляют поставки в каждый город от общего числа поставок детали P\*. Вывести город, число поставок деталей в этот город, общее число поставок детали P\*, процент.



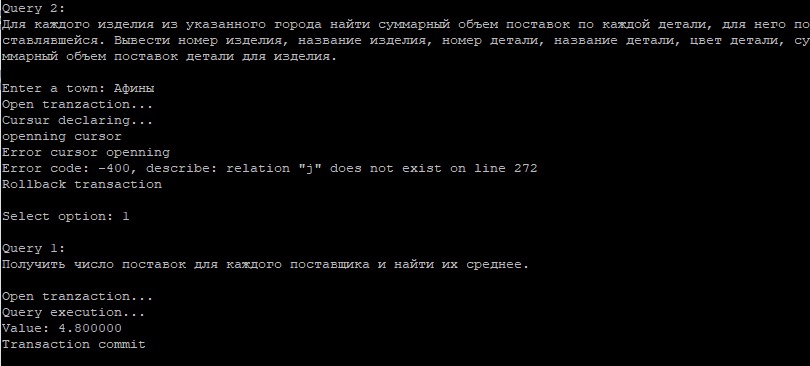
1. **Тесты**
   1. Передача неверного количества аргументов



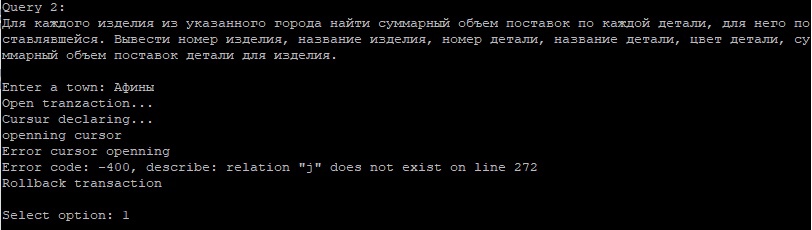
* 1. Отсутствие подходящих данных (запрос 3, вели деталь P10)



* 1. Проверка, что другие запросы работают после ошибки с одним из запросов



* 1. Обращение к несуществующей таблице (запрос 3)



1. **Текст программы**

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <sqltypes.h>

#include <string.h>

#include <sqlca.h>

#include <sqlda.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+1] = DEFAULT\_USER;

char password[PWD\_LEN+1] = DEFAULT\_PWD;

char schema[SCHEMA\_LEN+1] = DEFAULT\_SCHEMA;

char town\_name[30];

char enter\_n\_det[6];

float avg\_post;

char n\_det[6], n\_izd[6];

char det\_name[30], izd\_name[30];

char cvet[30];

int kol\_sum;

char town[30];

int kol\_post;

int obch\_kol\_post;

float procent;

char begin\_str[] = "BEGIN WORK";

char commit\_str[] = "COMMIT WORK";

char rollback\_str[] = "ROLLBACK WORK";

FROM (SELECT COUNT(\*) post\_count FROM spj GROUP BY spj.n\_post) t;";

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) Display options\n");

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

}

void close\_programm () {

EXEC SQL DEALLOCATE prepared\_query\_1;

EXEC SQL DEALLOCATE prepared\_query\_2;

EXEC SQL DEALLOCATE prepared\_query\_3;

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\n", errnum, sqlca.sqlerrm.sqlerrmc);

}

int main (int argc, char\* argv[]) {

if( argc != 1 && argc != 4){

printf("Parameters number != 3\nUse: ./lab3 <username> <password> <schema>\n");

return(EXIT\_FAILURE);

}

if( argc == 4){

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

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

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

}

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);

}

prepare\_tsql();

prepare\_queries();

EXEC SQL begin work; // начать транзакцию

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

EXEC SQL commit work; // закончить транзацию

int cmd = 1;

display\_options();

while (cmd != 0) {

printf("\nSelect option: ");

scanf("%d", &cmd);

switch(cmd) {

case 1:

query\_1();

break;

case 2:

query\_2();

break;

case 3:

query\_3();

break;

case 4:

display\_options();

break;

case 0:

close\_programm();

break;

default:

printf("\n");

break;

}

}

return 0;

}

void prepare\_queries() {

EXEC SQL BEGIN DECLARE SECTION;

char stmt\_1[] = "SELECT AVG(t.post\_count)\

FROM (SELECT COUNT(\*) post\_count\

FROM spj\

GROUP BY spj.n\_post) t;";

char stmt\_2[] ="SELECT t.n\_izd, i.name, t.n\_det, d.name, d.cvet, t.kol\_sum\

FROM (SELECT j.n\_izd, spj.n\_det, SUM(spj.kol) kol\_sum\

FROM j\

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

WHERE j.town = ?\

GROUP BY j.n\_izd, spj.n\_det) t\

JOIN j i ON i.n\_izd = t.n\_izd\

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

ORDER BY t.n\_izd, t.n\_det;";

char stmt\_3[] ="SELECT J.town, COUNT(\*) AS Kol\_Post,(SELECT COUNT(\*)\

FROM SPJ\

WHERE n\_det = ?) AS Obch\_kol\_post,\

COUNT(\*) \* 100.0 / (SELECT COUNT(\*)\

FROM SPJ\

WHERE n\_det = ?) AS Procent\

FROM J\

JOIN SPJ ON J.n\_izd = SPJ.n\_izd\

WHERE SPJ.n\_det = ?\

GROUP BY J.town;";

EXEC SQL END DECLARE SECTION;

EXEC SQL PREPARE prepared\_query\_1 FROM :stmt\_1;

EXEC SQL PREPARE prepared\_query\_2 FROM :stmt\_2;

EXEC SQL PREPARE prepared\_query\_3 FROM :stmt\_3;

}

void query\_1() {

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

printf("Получить число поставок для каждого поставщика и найти их среднее.\n\n");

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

EXEC SQL begin work;

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

EXEC SQL EXECUTE prepared\_query\_1 INTO :avg\_post;

if (sqlca.sqlcode < 0) {

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

printf("Rollback transaction\n");

EXEC SQL rollback work; // отменяем транзакцию

return;

}

printf("Value: %f \n", avg\_post);

printf("Transaction commit\n");

EXEC SQL commit work;

//if(sqlca.sqlcode < 0)

//{

// log\_error("Transaction begining", sqlca.sqlcode);

// printf("Rollback transaction\n");

// EXEC SQL rollback work;

//}

return;

}

void query\_2() {

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

printf("Для каждого изделия из указанного города найти суммарный объем поставок по каждой детали, для него поставлявшейся. Вывести номер изделия, название изделия, номер детали, название детали, цвет детали, суммарный объем поставок детали для изделия.\n\n");

printf("Enter a town: ");

scanf("%s", &town\_name);

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

EXEC SQL begin work;

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

EXEC SQL DECLARE cursor\_2 CURSOR FOR prepared\_query\_2;

// 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\_2 USING :town\_name;

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\_2 INTO :n\_izd, :izd\_name, :n\_det, :det\_name, :cvet, :kol\_sum;

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\_izd\tizd\_name\tn\_det\tdet\_name\tcvet\tkol\_sum\n");

printf("%s\t%s\t%s\t%s\t%s\t%d\n", n\_izd, izd\_name, n\_det, det\_name, cvet, kol\_sum);

row\_count++;

while(sqlca.sqlcode == 0){

EXEC SQL FETCH cursor\_2 INTO :n\_izd, :izd\_name, :n\_det, :det\_name, :cvet, :kol\_sum;

if (sqlca.sqlcode == 0) {

printf("%s\t%s\t%s\t%s\t%s\t%d\n", n\_izd, izd\_name, n\_det, det\_name, cvet, kol\_sum);

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 query\_3() {

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

printf("Ввести номер детали P\*. Найти города, в которые поставлялась деталь P\*, и определить, какой процент составляют поставки в каждый город от общего числа поставок детали P\*. Вывести город, число поставок деталей в этот город, общее число поставок детали P\*, процент. \n\n");

printf("Enter a detail: ");

scanf("%s", &n\_det);

printf(n\_det);

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

EXEC SQL begin work;

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

EXEC SQL DECLARE cursor\_3 CURSOR FOR prepared\_query\_3;

// 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\_3 USING :n\_det, :n\_det, :n\_det;

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\_3 INTO :town, :kol\_post, :obch\_kol\_post, :procent;

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("town\tkol\_poat\tobch\_kol\_post\tprocent\n");

printf("%s\t%d\t%d\t%f\n", town, kol\_post, obch\_kol\_post, procent);

row\_count++;

while(sqlca.sqlcode == 0){

EXEC SQL FETCH cursor\_3 INTO :town, :kol\_post, :obch\_kol\_post, :procent;

if (sqlca.sqlcode == 0) {

printf("%s\t%d\t%d\t%f\n", town, kol\_post, obch\_kol\_post, procent);

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;

}

}

}