|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 5 | | |
| по дисциплине «Технологии баз данных» | | |
| **Динамический SQL** | | |
|  | | |
|  | Бригада 10 | Субботин дмитрий |
| Группа ПМИ-12 | Швадченко артем |
| Вариант 6 |  |
|  |  |
|  |  |
|  |  |
| Преподаватели | Стасышина Татьяна леонидовна СИВАк мария алексеевна |
|  |  |
| Новосибирск, 2024 | | |

1. **Цель работы**

Приобрести практические навыки работы со средствами динамического SQL при написании программ на ESQL/C.

1. **Задания**

1. Получить для каждого поставщика средний вес поставки и найти их сумму.

2. Для указанной детали (P3) найти средний вес поставки для каждого изделия, для которого деталь поставлялась. Вывести номер изделия, название изделия, город изделия, средний вес поставки.

3. Ввести номер детали P\*. Найти изделия, для которых поставлялась деталь P\* , и определить, какой процент составляет объем поставок для каждого изделия от общего объема поставок детали P\*. Вывести номер изделия, объем поставок для этого изделия, общий объем поставок детали P\*, процент.

1. **Ход работы**

**Создание программы**

Необходимые программы компилируются с помощью команды **pgcci** из исходного кода **LR4.ec.** Для запуска последней необходимо прописать имя полученной программы и затем номер выполняемого запроса.

Пример:

./LR4.exe

2

**3.1 SQL-скрипты для заданий**

**№1:**

Получить для каждого поставщика средний вес поставки и найти их сумму.

Средний вес поставок каждого из поставщиков:

SELECT DISTINCT spj.n\_post, avg(spj.kol\*p.ves) AS averege

FROM spj

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

GROUP BY spj.n\_post

*Результат работы:*

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Сумма средних весов поставок каждого из поставщиков:

SELECT SUM(average) AS total

FROM (SELECT s.n\_post, AVG(p.ves \* spj.kol) AS average

FROM s

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

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

GROUP BY s.n\_post) AS sub

*Результат работы:*



**№2:**

Для указанной детали (P3) найти средний вес поставки для каждого изделия, для которого деталь поставлялась. Вывести номер изделия, название изделия, город изделия, средний вес поставки.

SELECT j.n\_izd, j.name, j.town, AVG(p.ves \* spj.kol) AS average

FROM j

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

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

WHERE p.n\_det = 'P3'

GROUP BY j.n\_izd, j.name, j.town

*Результат работы:*

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

**№3:**

Ввести номер детали P1. Найти изделия, для которых поставлялась деталь P1, и определить, какой процент составляет объем поставок для каждого изделия от общего объема поставок детали P1. Вывести номер изделия, объем поставок для этого изделия, общий объем поставок детали P1, процент.

SELECT DISTINCT n\_izd, sum(kol), (SELECT SUM(kol)

FROM spj

WHERE n\_det='P1') AS total, sum(kol)\*100/(SELECT SUM(kol)

FROM spj

WHERE n\_det='P1') AS percent

FROM spj

WHERE n\_det='P1'

GROUP BY n\_izd

*Результат работы:*

*Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание*

**3.2 Тесты ESQL/C программы**

**№1:**

Получить для каждого поставщика средний вес поставки и найти их сумму.

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

**№2:**

Для указанной детали (P3) найти средний вес поставки для каждого изделия, для которого деталь поставлялась. Вывести номер изделия, название изделия, город изделия, средний вес поставки.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

**№3:**

Ввести номер детали P\*. Найти изделия, для которых поставлялась деталь P\* , и определить, какой процент составляет объем поставок для каждого изделия от общего объема поставок детали P\*. Вывести номер изделия, объем поставок для этого изделия, общий объем поставок детали P\*, процент.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

*Тест с некорректным вводом n\_det в (2):*

*Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание*

*Тест с некорректным вводом n\_det в (3):*

*Изображение выглядит как текст, снимок экрана, Шрифт

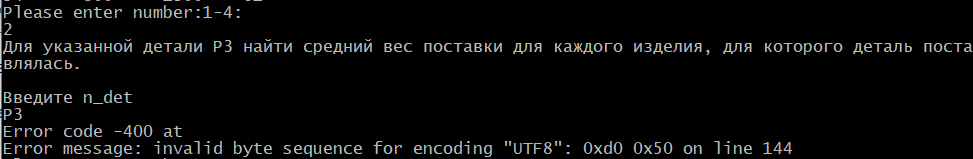
Автоматически созданное описание*

*Тест на правильность работы программы:*

*Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание*

*Удаляем поле таблицы P и пробуем выполнить(2):*

**

*Удаляем таблицу SPJ и пробуем выполнить(3):*

*Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание*

**3.3 ESQL/C-код программы**

#include <sqlca.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

void do\_error(int errnum)

{

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

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

    return;

}

bool Connect()

{

    //подключение к бд

    EXEC SQL connect to students@fpm2.ami.nstu.ru user "pmi-b1610" using "rti9hGsA\*v";

    if (sqlca.sqlcode != 0)

    {

        do\_error(sqlca.sqlcode);

            return false;

    }

    // схема бд

    EXEC SQL set search\_path to pmib1610;

    if (sqlca.sqlcode != 0)

    {

        do\_error(sqlca.sqlcode);

        return false;

    }

    return true;

}

int main()

{

    setlocale(LC\_ALL, "Russian");

    printf("Start\n");

    if (!Connect())//выполняем соединение с бд

    {

        printf("Connection error\nsqlcode=%d\n", sqlca.sqlcode);

        return 1;

    }

    printf("Connect\n");

    int n;

    bool flag = true;

    //в этой области задаем переменные sql (можно знаком $)

    EXEC SQL begin declare section;

    //1

    float summ;

    //2

    char n\_det[6], n\_izd[6], name[20], town[20];

    float kol;

    //3

    char n\_det2[6], n\_izd2[6];

    int pkol, okol, proc;

    EXEC SQL end declare section;

    //начало работы

    EXEC SQL begin work;

    // запрос summ - переменная инициализированная ранее. В ней будет лежать ре-зультат запроса

    EXEC SQL PREPARE select1 AS

        SELECT SUM(average)

        FROM (SELECT s.n\_post, CAST(AVG(p.ves \* spj.kol) as FLOAT) as average

             FROM s

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

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

             GROUP BY s.n\_post) AS sub;

    if (sqlca.sqlcode != 0)

    {

        do\_error(sqlca.sqlcode);

        EXEC SQL rollback work;// если ошибка откатываем запрос

    }

    ///////////////////////////////////////////////////////////////////////////////////////

    EXEC SQL PREPARE select2 AS

        SELECT j.n\_izd, j.name, j.town, CAST(AVG(p.ves \* spj.kol) as FLOAT)

        FROM j

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

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

        WHERE p.n\_det =?

        GROUP BY j.n\_izd, j.name, j.town;

    if (sqlca.sqlcode != 0)

    {

        do\_error(sqlca.sqlcode);

        EXEC SQL rollback work;// если ошибка откатываем запрос

    }

    //////////////////////////////////////////////////////////////////////////////////////

    EXEC SQL PREPARE selectsql AS

        SELECT DISTINCT n\_izd, sum(kol),

                        (SELECT SUM(kol)

                        FROM spj

                        WHERE n\_det=?), sum(kol)\*100/(SELECT SUM(kol)

                                                                 FROM spj

                                                                 WHERE n\_det=?)

        FROM spj

        WHERE n\_det=?

        GROUP BY n\_izd;

    ///////////////////////////////////////////////////////////////////////////////////

    do

    {

        printf("Please enter number:1-4:\n");

        scanf("%d", &n);

        switch (n)

        {

        case 1:

            printf("Получить для каждого поставщика средний вес поставки и найти их сумму. \n");

            EXEC SQL DECLARE cur1 cursor for select1;

            EXEC SQL OPEN cur1;

            if (sqlca.sqlcode != 0)

            {

                do\_error(sqlca.sqlcode);

                EXEC SQL rollback work;// если ошибка откатываем запрос

                break;

            }

            EXEC SQL FETCH cur1 into :summ;

            if (sqlca.sqlcode != 0)

            {

                do\_error(sqlca.sqlcode);

                break;

            }

            printf("Sum: %f\n", summ);

            EXEC SQL close cur1;

            if (sqlca.sqlcode != 0)

            {

                do\_error(sqlca.sqlcode);

                break;

            }

            EXEC SQL commit work; // сохраняем результат запроса

            break;

        case 2:

            printf("Для указанной детали P3 найти средний вес поставки для каждого изделия, для которого деталь поставлялась.\n");

            printf("\nВведите n\_det\n");

            scanf("%5s", n\_det);

            EXEC SQL DECLARE cur2 CURSOR FOR select2;

            if (sqlca.sqlcode != 0)

            {

                do\_error(sqlca.sqlcode);

                EXEC SQL rollback work;

                break;

            }

            EXEC SQL begin work;

            EXEC SQL OPEN cur2 using :n\_det;

            if (sqlca.sqlcode != 0)

            {

                do\_error(sqlca.sqlcode);

                break;

            }

            EXEC SQL FETCH NEXT FROM cur2 into :n\_izd, :name, :town, :kol;

            if (sqlca.sqlcode == 100)//если в результате запроса нет данных

            {

                printf("Данных не найдено\n");

                EXEC SQL close cur2;

                EXEC SQL commit work;

                break;

            }

            else if (sqlca.sqlcode < 0)//ошибка времени выполнения

            {

                do\_error(sqlca.sqlcode);

                EXEC SQL close cur2;

                EXEC SQL rollback work;

                break;

            }

            printf("%-6s%-6s%-20s%-20s\n", "n\_izd", "name", "town", "kol");

            do

            {

                printf("%s\t%s\t%s\t%f\n", n\_izd, name, town, kol);

                EXEC SQL FETCH NEXT from cur2 into :n\_izd, :name, :town, :kol;

                if (sqlca.sqlcode < 0)

                {

                    do\_error(sqlca.sqlcode);

                    EXEC SQL rollback work;

                    break;

                }

                //printf("%s\t%s\t%s\t%d\n", n\_izd, town, name, kol);

            } while (sqlca.sqlcode != 100);

            EXEC SQL CLOSE cur2;

            if (sqlca.sqlcode != 0)

            {

                do\_error(sqlca.sqlcode);

                break;

            }

            printf("Success!\n");

            EXEC SQL commit work;

            break;

        case 3:

            printf("Найти изделия, для которых поставлялась деталь P1, и определить, какой процент составляет объем поставок для каждого изделия от общего объема поставок детали P1 \n");

            printf("Введите n\_det\n");

            scanf("%5s", n\_det2);

            EXEC SQL DECLARE cur3 CURSOR FOR selectsql;

            if (sqlca.sqlcode != 0)

            {

                do\_error(sqlca.sqlcode);

                break;

            }

            EXEC SQL begin work;

            //открываем курсор

            EXEC SQL OPEN cur3 using :n\_det2, :n\_det2, :n\_det2;

            if (sqlca.sqlcode != 0)

            {

                do\_error(sqlca.sqlcode);

                EXEC SQL rollback work;

                EXEC SQL close cur3;

                break;

            }

            //начало работы

            EXEC SQL FETCH NEXT FROM cur3 into :n\_izd2, :pkol, :okol, :proc;

            if (sqlca.sqlcode == 100)//если в результате запроса нет данных

            {

                printf("Данных не найдено\n");

                EXEC SQL close cur3;

                EXEC SQL commit work;

                break;

            }

            else if (sqlca.sqlcode < 0)//ошибка времени выполнения

            {

                do\_error(sqlca.sqlcode);

                EXEC SQL close cur3;

                EXEC SQL rollback work;

                break;

            }

            printf("n\_izd\tpkol\tokol\tproc\n");

            do//повторяем пока в результате не будет данных

            {

                printf("%s\t%d\t%d\t%d\n", n\_izd2, pkol, okol, proc);

                EXEC SQL fetch next from cur3 into :n\_izd2, :pkol, :okol, :proc;

                if (sqlca.sqlcode < 0)

                {

                    do\_error(sqlca.sqlcode);

                    EXEC SQL close cur3;

                    EXEC SQL rollback work;

                    break;

                }

            } while (sqlca.sqlcode != 100);

            //закрытие курсора

            EXEC SQL CLOSE cur3;

            if (sqlca.sqlcode != 0)

            {

                do\_error(sqlca.sqlcode);

                break;

            }

            EXEC SQL commit work;

            break;

        case 4:

            flag = false;

            EXEC SQL disconnect all;

            printf("Session closed\n");

            break;

        default:

            printf("Wrong number!\n");

            break;

        }

    } while (flag);

    EXEC SQL disconnect all;//закрытие соединения с бд

    return 0;

}