|  |  |  |
| --- | --- | --- |
|  | | |
| 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** | | |
|  | | |
|  | 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 встроенным SQL средствами ESQL/C, включая использование курсоров и средств динамического SQL.

1. **Задания**

1. Выдать число поставщиков, поставлявших детали для изделий, собираемых в городе, где производят красные детали.

2. Поменять местами названия первого и последнего по алфавиту изделия, т. е. первому по алфавиту изделию установить название последнего по алфавиту изделия и наоборот.

3. Найти поставщиков, имеющих поставки, объем которых меньше объема наименьшей поставки красных деталей, сделанной этим поставщиком. Вывести номер поставщика, объем поставки, минимальный объем поставки красных деталей поставщиком.

4. Выбрать детали, не поставлявшиеся ни одним поставщиком, поставлявшим детали для изделия J3.

5. Выдать полную информацию об изделиях, у которых есть поставки ТОЛЬКО с весом от 1000 до 7000.

1. **Ход работы**

**Создание программы**

Необходимые программы компилируются с помощью команды **pgcci** из исходного кода **LR3.ec.** Для запуска последней необходимо прописать имя полученной программы и затем номер выполняемого запроса.

Пример:

./LR3.exe

2

**3.1 SQL-скрипты для заданий**

**№1:**

Выдать число поставщиков, поставлявших детали для изделий, собираемых в городе, где производят красные детали.

SELECT count(DISTINCT n\_post)

FROM spj

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

WHERE j.town IN (SELECT DISTINCT town

FROM p

WHERE cvet='Красный')

*Результат работы:*



**№2:**

Поменять местами названия первого и последнего по алфавиту изделия, т. е. первому по алфавиту изделию установить название последнего по алфавиту изделия и наоборот.

UPDATE j

SET name=CASE

WHEN name=(SELECT name

FROM j

ORDER BY name

LIMIT 1) THEN (SELECT name

FROM j

ORDER BY name DESC

LIMIT 1)

WHEN name=(SELECT name

FROM j

ORDER BY name DESC

LIMIT 1) THEN (SELECT name

FROM j

ORDER BY name

LIMIT 1)

END

WHERE (name=(SELECT name

FROM j

ORDER BY name

LIMIT 1))

OR (name=(SELECT name

FROM j

ORDER BY name DESC

LIMIT 1))

*Результат работы:*

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание



**№3:**

Найти поставщиков, имеющих поставки, объем которых меньше объема наименьшей поставки красных деталей, сделанной этим поставщиком. Вывести номер поставщика, объем поставки, минимальный объем поставки красных деталей поставщиком.

SELECT spj.n\_post, spj.kol, k.kk

FROM spj

LEFT JOIN (SELECT n\_post, min(kol) AS kk

FROM spj

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

WHERE p.cvet='Красный'

GROUP BY spj.n\_post) AS k ON spj.n\_post=k.n\_post

WHERE spj.kol<k.kk

*Результат работы:*

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

Автоматически созданное описание*

**№4:**

Выбрать детали, не поставлявшиеся ни одним поставщиком, поставлявшим детали для изделия J3.

SELECT DISTINCT n\_det

FROM spj

EXCEPT

SELECT DISTINCT n\_det

FROM spj

WHERE n\_post IN (SELECT DISTINCT n\_post

FROM spj

WHERE n\_izd='J3')

*Результат работы:*

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

Автоматически созданное описание*

**№5:**

Выдать полную информацию об изделиях, у которых есть поставки ТОЛЬКО с весом от 1000 до 7000.

SELECT \*

FROM j

WHERE n\_izd IN (SELECT DISTINCT n\_izd

FROM spj

EXCEPT

SELECT DISTINCT n\_izd

FROM spj

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

WHERE (spj.kol\*p.ves) NOT BETWEEN 1000 AND 7000)

*Результат работы:*

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

Автоматически созданное описание*

**3.2 Тесты ESQL/C программы**

**№1:**

Выдать число поставщиков, поставлявших детали для изделий, собираемых в городе, где производят красные детали.

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

**№2:**

Поменять местами названия первого и последнего по алфавиту изделия, т. е. первому по алфавиту изделию установить название последнего по алфавиту изделия и наоборот.



**№3:**

Найти поставщиков, имеющих поставки, объем которых меньше объема наименьшей поставки красных деталей, сделанной этим поставщиком. Вывести номер поставщика, объем поставки, минимальный объем поставки красных деталей поставщиком.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

**№4:**

Выбрать детали, не поставлявшиеся ни одним поставщиком, поставлявшим детали для изделия J3.

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

**№5:**

Выдать полную информацию об изделиях, у которых есть поставки ТОЛЬКО с весом от 1000 до 7000.

**Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание**

*Тест с нахождением несуществующей строки:*

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

Автоматически созданное описание*

*Тест со значением NULL полей j.name и j.town:*

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

Автоматически созданное описание*

*Ошибка в declare section “Variables declared”:*

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

Автоматически созданное описание*

*Ошибка “Cursor opened”:*

*Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание*

*Ошибка редактирования из-за создания ограничения*

*alter table j add constraint che check(n\_izd!='J1' or name!='Жесткий диск'):*

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**3.3 ESQL/C-код программы**

#define DB "students@students.ami.nstu.ru"

#define WARN 1

#define NONE 0

#include <stdlib.h>

const char\* DEFAULT\_LOGIN = "pmi-b1610";

const char\* DEFAULT\_PASSWORD = "rti9hGsA\*v";

const char\* DEFAULT\_SCHEMA = "pmib1610";

void log\_errors(const char\* operation) {

    fprintf(stderr, "error: {%d} on %s\n", sqlca.sqlcode,operation);

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

}

void handle\_errors(const char\* operation) {

    if (sqlca.sqlcode < 0) {

        log\_errors(operation);

        exit(EXIT\_FAILURE);

    }

}

void log\_warnings(const char\* operation) {

    fprintf(stderr, "warning: {%d} on %s\n", sqlca.sqlcode,operation);

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

}

int handle\_warnings(const char\* operation) {

    if (sqlca.sqlcode < 0) {

        log\_warnings(operation);

        return WARN;

    }

    return NONE;

}

void connect\_database(const char\* login, const char\* password) {

    EXEC SQL BEGIN DECLARE SECTION;

        const char\* sql\_login = login;

        const char\* sql\_password = password;

        const char\* database = DB;

    EXEC SQL END DECLARE SECTION;

    EXEC SQL CONNECT TO :database USER :sql\_login USING :sql\_password;

    handle\_errors("Databaseconnection");

}

void connect\_schema(const char\* name) {

    EXEC SQL BEGIN DECLARE SECTION;

        const char\* schema\_path = name;

    EXEC SQL END DECLARE SECTION;

    EXEC SQL SET search\_path TO :schema\_path;

    handle\_errors("Set schema");

}

void first() {

    EXEC SQL BEGIN DECLARE SECTION;

        int count;

    EXEC SQL END DECLARE SECTION;

    if (handle\_warnings("Variables declared") == WARN) {

        return;

    }

    EXEC SQL BEGIN WORK;

    EXEC SQL

        SELECT COUNT(DISTINCT n\_det) INTO :count

        FROM spj

        JOIN j ON j.n\_izd=spj.n\_izd

        WHERE j.town IN (SELECT DISTINCT town

                        FROM p

                        WHERE cvet='Красный');

    if (handle\_warnings("Query finished") == WARN) {

        EXEC SQL ROLLBACK WORK;

    } else {

        EXEC SQL COMMIT WORK;

        printf("Count: %d\n", count);

    }

}

void second() {

    EXEC SQL BEGIN WORK;

    EXEC SQL

        UPDATE j

        SET name=CASE

            WHEN name=(SELECT name

                       FROM j

                       ORDER BY name

                       LIMIT 1) THEN (SELECT name

                                     FROM j

                                     ORDER BY name DESC

                                     LIMIT 1)

            WHEN name=(SELECT name

                       FROM j

                       ORDER BY name DESC

                       LIMIT 1) THEN (SELECT name

                                     FROM j

                                     ORDER BY name

                                     LIMIT 1)

        END

        WHERE (name=(SELECT name

                    FROM j

                    ORDER BY name

                    LIMIT 1))

        OR (name=(SELECT name

                 FROM j

                 ORDER BY name DESC

                 LIMIT 1));

if (sqlca.sqlcode < 0)

   {

      printf("Ошибка запроса update! Код: %d (%s)\n Текст ошибки %s\n",

         sqlca.sqlcode,

         sqlca.sqlstate,

         sqlca.sqlerrm.sqlerrmc);

      exec SQL rollback;

      return;

   }

    if (handle\_warnings("Query finished") == WARN) {

        EXEC SQL ROLLBACK WORK;

    } else {

        printf("Changed: %lld rows\n", sqlca.sqlerrd[2]);

        EXEC SQL COMMIT WORK;

    }

}

void third() {

    int rows\_count = 0;

    EXEC SQL BEGIN DECLARE SECTION;

        struct {

            char n\_post[2 \* 6 + 1];

            int kol;

            int kk;

        } data;

    EXEC SQL END DECLARE SECTION;

    if (handle\_warnings("Variables declared") == WARN) {

        return;

    }

    EXEC SQL DECLARE third CURSOR FOR

    SELECT spj.n\_post, spj.kol, k.kk

    FROM spj

    LEFT JOIN (SELECT n\_post, min(kol) AS kk

              FROM spj

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

              WHERE p.cvet='Красный'

              GROUP BY spj.n\_post) AS k ON spj.n\_post=k.n\_post

    WHERE spj.kol<k.kk;

    if (handle\_warnings("Cursor declared") == WARN) {

        EXEC SQL ROLLBACK WORK;

        return;

    }

    EXEC SQL OPEN third;

    if (handle\_warnings("Cursor opened") == WARN) {

        EXEC SQL ROLLBACK WORK;

        return;

    }

    EXEC SQL FETCH NEXT third INTO :data.n\_post, :data.kol, :data.kk;

    if (handle\_warnings("Fetch next") == WARN) {

        EXEC SQL ROLLBACK WORK;

        return;

    }

    if (sqlca.sqlcode == 100) {

        printf("No data\n");

    }

    if (sqlca.sqlcode == 0) {

        rows\_count++;

        printf("n\_post\tkol\tkk\n");

        printf("%s\t%d\t%d\n", data.n\_post, data.kol, data.kk);

    }

    while (sqlca.sqlcode == 0) {

        EXEC SQL FETCH NEXT third INTO :data.n\_post, :data.kol, :data.kk;

        if (handle\_warnings("Fetch next") == WARN) {

            EXEC SQL ROLLBACK WORK;

            return;

        }

        if (sqlca.sqlcode == 0) {

            rows\_count++;

            printf("%s\t%d\t%d\n", data.n\_post, data.kol, data.kk);

        }

    }

    EXEC SQL CLOSE third;

    if (handle\_warnings("Query finished") == WARN) {

        EXEC SQL ROLLBACK WORK;

    } else {

        EXEC SQL COMMIT WORK;

        printf("Found: %d rows\n", rows\_count);

    }

}

void fourth() {

    int rows\_count = 0;

    EXEC SQL BEGIN DECLARE SECTION;

        char n\_det[2 \* 6 + 1];

    EXEC SQL END DECLARE SECTION;

    if (handle\_warnings("Variables declared") == WARN) {

        return;

    }

    EXEC SQL DECLARE fourth CURSOR FOR

    SELECT DISTINCT n\_det

    FROM spj

    EXCEPT

    SELECT DISTINCT n\_det

    FROM spj

    WHERE n\_post IN (SELECT DISTINCT n\_post

                    FROM spj

                    WHERE n\_izd='J3');

    if (handle\_warnings("Cursor declared") == WARN) {

        EXEC SQL ROLLBACK WORK;

        return;

    }

    EXEC SQL OPEN fourth;

    if (handle\_warnings("Cursor opened") == WARN) {

        EXEC SQL ROLLBACK WORK;

        return;

    }

    EXEC SQL FETCH fourth INTO :n\_det;

    if (handle\_warnings("Fetch next") == WARN) {

        EXEC SQL ROLLBACK WORK;

        return;

    }

    if (sqlca.sqlcode == 100) {

        printf("No data\n");

    }

    if (sqlca.sqlcode == 0){

        rows\_count++;

        printf("n\_det\n");

        printf("%s\n", n\_det);

    }

    while (sqlca.sqlcode == 0) {

        EXEC SQL FETCH fourth INTO :n\_det;

        if (handle\_warnings("Fetch next") == WARN) {

            EXEC SQL ROLLBACK WORK;

            return;

        }

        if (sqlca.sqlcode == 0) {

            rows\_count++;

            printf("%s\n", n\_det);

        }

    }

    EXEC SQL CLOSE fourth;

    if (handle\_warnings("Query finished") == WARN) {

        EXEC SQL ROLLBACK WORK;

    } else {

        EXEC SQL COMMIT WORK;

        printf("Found: %d rows\n", rows\_count);

    }

}

void fifth() {

    int rows\_count = 0;

    EXEC SQL BEGIN DECLARE SECTION;

    struct {

        char n\_izd[2 \* 6 + 1];

        char name[2 \* 20 + 1];

        char town[2 \* 20 + 1];

    } part;

    short indna, indto;

    EXEC SQL END DECLARE SECTION;

    if (handle\_warnings("Variables declared") == WARN) {

        return;

    }

    EXEC SQL DECLARE fifth CURSOR FOR

    SELECT \*

    FROM j

    WHERE n\_izd IN (SELECT DISTINCT n\_izd

                   FROM spj

                   EXCEPT

                   SELECT DISTINCT n\_izd

                   FROM spj

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

                   WHERE (spj.kol\*p.ves) NOT BETWEEN 1000 AND 7000);

    if (handle\_warnings("Cursor declared") == WARN) {

        EXEC SQL ROLLBACK WORK;

        return;

    }

    EXEC SQL OPEN fifth;

    if (handle\_warnings("Cursor opened") == WARN) {

        EXEC SQL ROLLBACK WORK;

        return;

    }

    EXEC SQL FETCH NEXT fifth INTO :part.n\_izd, :part.name INDICATOR indna, :part.town INDICATOR indto;

    if (handle\_warnings("Fetch next") == WARN) {

        EXEC SQL ROLLBACK WORK;

        return;

    }

    if (sqlca.sqlcode == 100) {

        printf("No data\n");

    }

    if (sqlca.sqlcode == 0) {

        rows\_count++;

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

        if(indna<0)

        printf("%-6s%-20s%-20s\n", part.n\_izd, "NULL", part.town);

        else if(indto<0)

        printf("%-6s%-20s%-20s\n", part.n\_izd, part.name, "NULL");

        else

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

    }

    while (sqlca.sqlcode == 0) {

        EXEC SQL FETCH NEXT fifth INTO :part.n\_izd, :part.name INDICATOR indna, :part.town INDICATOR indto;

        if (handle\_warnings("Fetch next") == WARN) {

            EXEC SQL ROLLBACK WORK;

            return;

        }

        if (sqlca.sqlcode == 0) {

            rows\_count++;

            if(indna<0)

            printf("%-6s%-20s%-20s\n", part.n\_izd, "NULL", part.town);

            else if(indto<0)

            printf("%-6s%-20s%-20s\n", part.n\_izd, part.name, "NULL");

            else

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

        }

    }

    EXEC SQL CLOSE fifth;

    if (handle\_warnings("Query finished") == WARN) {

        EXEC SQL ROLLBACK WORK;

    } else {

        EXEC SQL COMMIT WORK;

        printf("Found: %d rows\n", rows\_count);

    }

}

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

    const char\* login;

    const char\* password;

    const char\* schema;

    if (argc == 4) {

        login = argv[1];

        password = argv[2];

        schema = argv[3];

    } else {

        login = DEFAULT\_LOGIN;

        password = DEFAULT\_PASSWORD;

        schema = DEFAULT\_SCHEMA;

    }

    connect\_database(login, password);

    connect\_schema(schema);

    int exit = false;

    int task;

    printf("Enter: (1-5)-complete the task, (0)-exit\n");

    while(exit == false) {

        printf("=> ");

        scanf("%d", &task);

        switch (task) {

            case 0:

                exit = true;

                break;

            case 1:

                first();

                break;

            case 2:

                second();

                break;

            case 3:

                third();

                break;

            case 4:

                fourth();

                break;

            case 5:

                fifth();

                break;

            default:

                printf("Enter: (1-5)-complete the task, (0)-exit\n");

                break;

        }

        printf("\n");

    }

    return;

}