



МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Новосибирский государственный технический университет»



**НГТУ
НЭТИ** | **Факультет прикладной
математики и информатики**

Кафедра теоретической и прикладной информатики

Лабораторная работа 3
по дисциплине «Технологии баз данных»

РАБОТА С БАЗОЙ ДАННЫХ СРЕДСТВАМИ ВСТРОЕННОГО SQL

Группа	ПМ-92
Бригада	pmi-b9209
Вариант	4
Студенты	ИВАНОВ ВЛАДИСЛАВ КУТУЗОВ ИВАН
Преподаватели	СТАСЫШИНА Т. Л. СИВАК М. А.

Новосибирск, 2022

1 SQL-запросы

1. Выдать число деталей, которые поставлялись поставщиками, имеющими поставки с объемом от 600 до 700 деталей.

```
SELECT COUNT(DISTINCT n_det)
FROM spj
WHERE n_post IN (SELECT n_post
                  FROM spj
                  GROUP BY n_post, kol
                  HAVING kol >= 600 AND kol <= 700)
```

count

3

1 row(s)

2. Поменять местами цвета самой тяжелой и самой легкой детали, т.е. деталям с наибольшим весом установить цвет детали с минимальным весом, а деталям с минимальным весом установить цвет детали с наибольшим весом. Если цветов несколько, брать первый по алфавиту из этих цветов.

```
UPDATE p
SET cvet = (CASE WHEN ves = (SELECT MIN(ves) FROM p)
                  THEN (SELECT MIN(cvet) FROM p WHERE ves = (SELECT MAX(ves) FROM p))
                  ELSE (SELECT MIN(cvet) FROM p WHERE ves = (SELECT MIN(ves) FROM p))
                  END)
WHERE (ves IN (SELECT MIN(ves) FROM p)) OR (ves IN (SELECT MAX(ves) FROM p))
```

n_det	name	cvet	ves	town
P1	Гайка	Красный	12	Лондон
P2	Болт	Зеленый	17	Париж
P3	Винт	Голубой	17	Рим
P4	Винт	Красный	14	Лондон
P5	Кулачок	Голубой	12	Париж
P6	Блюм	Красный	19	Лондон

6 row(s)

n_det	name	cvet	ves	town
P2	Болт	Зеленый	17	Париж
P3	Винт	Голубой	17	Рим
P4	Винт	Красный	14	Лондон
P1	Гайка	Красный	12	Лондон
P5	Кулачок	Красный	12	Париж
P6	Блюм	Голубой	19	Лондон

6 row(s)

3. Найти поставщиков, имеющих поставки, вес которых составляет менее четверти наибольшего веса поставки этого поставщика. Вывести номер поставщика, вес поставки, четверть наибольшего веса поставки поставщика.

```
SELECT T1.n_post, (ves * kol) AS ves_post, max_ves_post_div_4
FROM (SELECT n_post, spj.n_det, kol, ves
      FROM spj JOIN p
      ON spj.n_det = p.n_det) AS T1 JOIN (SELECT n_post, MAX(ves * kol) / 4
      AS max_ves_post_div_4
      FROM spj JOIN p
      ON spj.n_det = p.n_det
      GROUP BY n_post) AS T2
ON T1.n_post = T2.n_post
WHERE ves * kol < max_ves_post_div_4
```

n_post	ves_post	max_ves_post_div_4
S2	1200	3400
S5	1700	2800
S5	1200	2800
S5	1200	2800

4 row(s)

4. Выбрать изделия, для которых не поставлялось ни одной из деталей, поставляемых поставщиком S4.

```
SELECT DISTINCT n_izd
FROM spj
EXCEPT
SELECT DISTINCT n_izd
FROM spj
WHERE n_det = (SELECT DISTINCT n_det
                FROM spj
                WHERE n_post = 'S4')
```

n_izd
J6
J5
J1

3 row(s)

5. Выдать полную информацию о деталях, которые поставлялись ТОЛЬКО поставщиками с максимальным рейтингом.

```
SELECT *
FROM p
WHERE n_det NOT IN (SELECT DISTINCT n_det
                    FROM spj
                    WHERE n_post NOT IN (SELECT n_post
                                          FROM s
                                          WHERE reiting = (SELECT MAX(reiting)
                                                            FROM s)))
```

n_det	name	cvet	ves	town
P2	Болт	Зеленый	17	Париж
P4	Винт	Красный	14	Лондон

2 row(s)

2 Код программы

2.1 Makefile

```
1 CC = /usr/bin/gcc
2 PGPATH = /usr/pgsql-9.3
3 ECPG = ${PGPATH}/bin/ecpg
4 LFLAGS = -L${PGPATH}/lib
5 CFLAGS = -I${PGPATH}/include
6 LIBS = -lecpG -lecpG_compat
7
8 TASKS_SRC := $(wildcard tasks.ec)
9 TASKS_OBJ := $(TASKS_SRC:tasks.ec=tasks.o)
10 TASKS_BIN := $(TASKS_SRC:tasks.ec=tasks~)
11
12 .INTERMEDIATE: $(TASKS_OBJ) db.o db.c
13
14 $(V).SILENT:
15
16 all: clean $(TASKS_BIN)
17
18 tasks~: tasks.o db.o
19     @echo -n "Compiling "$@"..."
20     $(CC) $(CFLAGS) -o $@ $< db.o $(LFLAGS) $(LIBS)
21     @echo " Done."
22
23 tasks.o: tasks.c
24     $(CC) $(CFLAGS) -c $< $(LFLAGS) $(LIBS)
25
26 tasks.c: tasks.ec
27     $(ECPG) -c $<
28
29 db.o: db.c
30     $(CC) $(CFLAGS) -c $< $(LFLAGS) $(LIBS)
31
32 db.c: db.ec
33     $(ECPG) -c $<
34
35 clean:
36     @echo "Cleaning up..."
37     rm -rf $(TASKS_BIN)
```

2.2 db.h

```
1 #define DB "students@students.ami.nstu.ru"
2
3 #define SOME_WARNING 1
```

```

4  #define NONE 0
5
6  void handle_errors(const char* operation);
7  void log_errors(const char* operation);
8  int handle_warnings(const char* operation);
9  void log_warnings(const char* operation);
10 void connect_database(const char* login, const char* password);
11 void connect_schema(const char* name);

```

2.3 db.ec

```

1  #include <stdlib.h>
2
3  #include "db.h"
4
5  void handle_errors(const char* operation) {
6      if (sqlca.sqlcode < 0) {
7          log_errors(operation);
8
9          exit(EXIT_FAILURE);
10     }
11 }
12
13 void log_errors(const char* operation) {
14     fprintf(stderr, "error code: {%d} on %s\n", sqlca.sqlcode,
15 ↪ operation);
16     fprintf(stderr, "message: %s\n", sqlca.sqlerrm.sqlerrmc);
17 }
18
19 int handle_warnings(const char* operation) {
20     if (sqlca.sqlcode < 0) {
21         log_warnings(operation);
22         return SOME_WARNING;
23     }
24
25     return NONE;
26 }
27
28 void log_warnings(const char* operation) {
29     fprintf(stderr, "warning code: {%d} on %s\n", sqlca.sqlcode,
30 ↪ operation);
31     fprintf(stderr, "message: %s\n", sqlca.sqlerrm.sqlerrmc);
32 }
33
34 void connect_database(const char* login, const char* password) {
35     EXEC SQL BEGIN DECLARE SECTION;
36     const char* sql_login = login;
37     const char* sql_password = password;

```

```

36     const char* database = DB;
37     EXEC SQL END DECLARE SECTION;
38
39     EXEC SQL CONNECT TO :database USER :sql_login USING :sql_password;
40     handle_errors("Database connection");
41 }
42
43 void connect_schema(const char* name) {
44     EXEC SQL BEGIN DECLARE SECTION;
45     const char* schema_path = name;
46     EXEC SQL END DECLARE SECTION;
47
48     EXEC SQL SET search_path TO :schema_path;
49     handle_errors("Set schema");
50 }

```

2.4 tasks.ec

```

1  #include "db.h"
2
3  const char* DEFAULT_LOGIN = "pmi-b9209";
4  const char* DEFAULT_PASSWORD = "mikilOj1";
5  const char* DEFAULT_SCHEMA = "pmib9209";
6
7  void first() {
8      EXEC SQL BEGIN DECLARE SECTION;
9      int count;
10     EXEC SQL END DECLARE SECTION;
11
12     if (handle_warnings("Variables declared") == SOME_WARNING) {
13         return;
14     }
15
16     EXEC SQL BEGIN WORK;
17     EXEC SQL
18         SELECT COUNT(DISTINCT n_det) INTO :count
19         FROM spj
20         WHERE n_post IN (
21             SELECT n_post
22             FROM spj
23             GROUP BY n_post, kol
24             HAVING kol >= 600 AND kol <= 700
25         );
26
27     int maybe_warning = handle_warnings("Query finished");
28
29     if (maybe_warning == SOME_WARNING) {
30         EXEC SQL ROLLBACK WORK;

```

```

31     } else {
32         EXEC SQL COMMIT WORK;
33         printf("count: %d\n", count);
34     }
35 }
36
37
38 void second() {
39     EXEC SQL BEGIN WORK;
40
41     EXEC SQL
42         UPDATE p SET cvet = (
43             CASE WHEN ves = (SELECT MIN(ves) FROM p)
44                 THEN (SELECT MIN(cvet) FROM p WHERE ves = (SELECT
↪ MAX(ves) FROM p))
45                 ELSE (SELECT MIN(cvet) FROM p WHERE ves = (SELECT
↪ MIN(ves) FROM p))
46             END
47         )
48         WHERE (ves IN (SELECT MIN(ves) FROM p)) OR (ves IN (SELECT
↪ MAX(ves) FROM p));
49
50     int maybe_warning = handle_warnings("Query finished");
51
52     if (maybe_warning == SOME_WARNING) {
53         EXEC SQL ROLLBACK WORK;
54     } else {
55         printf("Changed: %lld\n rows", sqlca.sqlerrd[2]);
56         EXEC SQL COMMIT WORK;
57     }
58 }
59
60 void third() {
61     int rows_count = 0;
62
63     EXEC SQL BEGIN DECLARE SECTION;
64     struct {
65         char n_post[2 * 6 + 1];
66         int ves;
67         int quarter_weight_of_delivery;
68     } data;
69     EXEC SQL END DECLARE SECTION;
70
71     if (handle_warnings("Variables declared") == SOME_WARNING) {
72         return;
73     }
74
75
76     EXEC SQL BEGIN WORK;

```

```

77 EXEC SQL DECLARE third CURSOR FOR
78     SELECT T1.n_post, (ves * kol) AS ves_post, max_ves_post_div_4
79     FROM (
80         SELECT n_post, spj.n_det, kol, ves
81         FROM spj JOIN p
82         ON spj.n_det = p.n_det
83     ) AS T1 JOIN (
84         SELECT n_post, MAX(ves * kol) / 4 AS max_ves_post_div_4
85         FROM spj JOIN p
86         ON spj.n_det = p.n_det
87         GROUP BY n_post
88     ) AS T2
89     ON T1.n_post = T2.n_post
90     WHERE ves * kol < max_ves_post_div_4;
91
92 EXEC SQL OPEN third;
93
94 if (handle_warnings("Cursor opened") == SOME_WARNING) {
95     return;
96 }
97
98 EXEC SQL FETCH NEXT third INTO :data.n_post, :data.ves,
99 ↪ :data.quarter_weight_of_delivery;
100
101 if (sqlca.sqlcode == 0) {
102     rows_count++;
103     printf("Поставщик\tВес поставки\tЧетверть\n");
104     printf("%s\t%d\t%d\n", data.n_post, data.ves,
105 ↪ data.quarter_weight_of_delivery);
106 }
107
108 while (sqlca.sqlcode == 0) {
109     EXEC SQL FETCH NEXT third INTO :data.n_post, :data.ves,
110 ↪ :data.quarter_weight_of_delivery;
111
112     if (sqlca.sqlcode == 0) {
113         rows_count++;
114         printf("%s\t%d\t%d\n", data.n_post, data.ves,
115 ↪ data.quarter_weight_of_delivery);
116     }
117 }
118
119 EXEC SQL CLOSE third;
120
121 int maybe_warning = handle_warnings("Query finished");
122
123 if (maybe_warning == SOME_WARNING) {
124     EXEC SQL ROLLBACK WORK;

```



```

122     } else {
123         EXEC SQL COMMIT WORK;
124         printf("Found: %d\n rows", rows_count);
125     }
126 }
127
128 void fourth() {
129     int rows_count = 0;
130
131     EXEC SQL BEGIN DECLARE SECTION;
132     char n_izd[2 * 6 + 1];
133     EXEC SQL END DECLARE SECTION;
134
135     if (handle_warnings("Variables declared") == SOME_WARNING) {
136         return;
137     }
138
139     EXEC SQL DECLARE fourth CURSOR FOR
140         SELECT DISTINCT n_izd
141         FROM spj
142         EXCEPT
143         SELECT DISTINCT n_izd
144         FROM spj
145         WHERE n_det = (SELECT DISTINCT n_det
146             FROM spj
147             WHERE n_post = 'S4'
148         );
149
150     EXEC SQL OPEN fourth;
151
152     if (handle_warnings("Cursor opened") == SOME_WARNING) {
153         return;
154     }
155
156     EXEC SQL FETCH fourth INTO :n_izd;
157
158     if (sqlca.sqlcode == 0) {
159         rows_count++;
160         printf("Изделие\n");
161         printf("%s\n", n_izd);
162     }
163
164     while (sqlca.sqlcode == 0) {
165         EXEC SQL FETCH fourth INTO :n_izd;
166
167         if (sqlca.sqlcode == 0) {
168             rows_count++;
169             printf("%s\n", n_izd);
170         }

```

```

171     }
172
173     EXEC SQL CLOSE fourth;
174
175     int maybe_warning = handle_warnings("Query finished");
176
177     if (maybe_warning == SOME_WARNING) {
178         EXEC SQL ROLLBACK WORK;
179     } else {
180         EXEC SQL COMMIT WORK;
181         printf("Found: %d\n rows", rows_count);
182     }
183 }
184
185 void fifth() {
186     int rows_count = 0;
187
188     EXEC SQL BEGIN DECLARE SECTION;
189     struct {
190         char n_det[2 * 6 + 1];
191         char name[2 * 20 + 1];
192         char cvet[2 * 20 + 1];
193         int ves;
194         char town[2 * 20 + 1];
195     } part;
196     EXEC SQL END DECLARE SECTION;
197
198     if (handle_warnings("Variables declared") == SOME_WARNING) {
199         return;
200     }
201
202     EXEC SQL DECLARE fifth CURSOR FOR
203         SELECT p.*
204         FROM p
205         WHERE n_det NOT IN (
206             SELECT DISTINCT n_det
207             FROM spj
208             WHERE n_post NOT IN (
209                 SELECT n_post
210                 FROM s
211                 WHERE reiting = (SELECT MAX(reiting) FROM s)
212             )
213         );
214
215     EXEC SQL OPEN fifth;
216
217     if (handle_warnings("Cursor opened") == SOME_WARNING) {
218         return;
219     }

```



```

265 connect_database(login, password);
266 connect_schema(schema);
267
268 int exit = false;
269 int task;
270
271 printf("Choose task number {1-5} or 0 to exit.\n");
272
273 while(exit == false) {
274     printf("=> ");
275     scanf("%d", &task);
276
277     switch (task) {
278         case 0:
279             exit = true;
280             break;
281
282         case 1:
283             first();
284             break;
285
286         case 2:
287             second();
288             break;
289
290         case 3:
291             third();
292             break;
293
294         case 4:
295             fourth();
296             break;
297
298         case 5:
299             fifth();
300             break;
301
302         default:
303             printf("Chose task number {1-5} or 0 to exit.\n");
304             break;
305     }
306
307     printf("\n");
308 }
309
310 return;
311 }

```

3 Результат выполнения

1. Выдать число деталей, которые поставлялись поставщиками, имеющими поставки с объемом от 600 до 700 деталей.

```
Choose task number {1-5} or 0 to exit.  
=> 1  
count: 3
```

2. Поменять местами цвета самой тяжелой и самой легкой детали, т.е. деталям с наибольшим весом установить цвет детали с минимальным весом, а деталям с минимальным весом установить цвет детали с наибольшим весом. Если цветов несколько, брать первый по алфавиту из этих цветов.

```
=> 2  
Changed: 3  
rows
```

3. Найти поставщиков, имеющих поставки, вес которых составляет менее четверти наибольшего веса поставки этого поставщика. Вывести номер поставщика, вес поставки, четверть наибольшего веса поставки поставщика.

```
=> 3  
Поставщик      Вес поставки      Четверть  
S2              1200              3400  
S5              1700              2800  
S5              1200              2800  
S5              1200              2800  
Found: 4  
rows
```

4. Выбрать изделия, для которых не поставлялось ни одной из деталей, поставляемых поставщиком S4.

```
=> 4  
Изделие  
J6  
J5  
J1  
Found: 3  
rows
```

5. Выдать полную информацию о деталях, которые поставлялись ТОЛЬКО поставщиками с максимальным рейтингом.

```
=> 5  
Номер  Наименование      Цвет      Вес      Город  
P2     Болт              Зеленый    17      Париж  
P4     Винт              Красный    14      Лондон  
Found: 2  
rows
```

6. Попытка выполнения п. 2 с заданным ограничением.

[Check constraint added.](#)

Name	Definition	Actions	Comment
p_n_det_key	UNIQUE (n_det)	Drop	
update-constraint	CHECK (name = 'Гайка'::bpchar AND cvet = 'Красный'::bpchar OR name <> 'Гайка'::bpchar)	Drop	

```
=> 2  
warning code: {-400} on Query finished  
message: new row for relation "p" violates check constraint "update-constraint" on line 41
```

7. Попытка выполнения программы после удаления таблиц.

j: Table dropped.
p: Table dropped.
s: Table dropped.
spj: Table dropped.

No tables found.

```
=> 1
warning code: {-400} on Query finished
message: permission denied for relation spj on line 17

=> 3
warning code: {-400} on Cursor opened
message: permission denied for relation spj on line 93
```

8. Попытка выполнения программы после удаления всех столбцов из таблицы spj.

```
=> 1
count: 0

=> 3
Found: 0
rows
=> 4
Found: 0
rows
```