ГУАП

КАФЕДРА № 42

ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ	[
ПРЕПОДАВАТЕЛЬ				
Доцент		Бржезовский А. В.		
должность, уч. степень, звание	подпись, дата	инициалы, фамилия		
	О ЛАБОРАТОРНОЙ РА			
ХРАНИМ	мые процедуры и (ФУНКЦИИ		
Вариант 5				
Management		_ 1		
по курсу: Методы и ср	едства проектирования и и технологий	нформационных систем		

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4128		Воробьев В.А.
, ,		подпись, дата	инициалы, фамилия

СОДЕРЖАНИЕ

1	Пос	тановка задачи	3
	1.1	Цель работы	3
	1.2	Задание	3
	1.3	Содержание отчета	3
2	Выі	полнение работы	4
	2.1	Вставка с пополнением справочников	4
	2.2	Удаление с очисткой справочников	5
	2.3	Каскадное удаление	7
	2.4	Вычисление и возврат значения агрегатной функции	8
	2.5	Формирование статистики во временной таблице	9
	2.6	Запросы использующие скалярную и табличную функцию	10
	2.7	Запросы с операторами	12
3	Вын	вод	14
Cl	ПИС	ОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	15
П	РИ.ЛО	ЭЖЕНИЕ	16

1 Постановка задачи

1.1 Цель работы

Получить навыки работы с хранимыми процедурами, хранимыми функциями и операторами.

1.2 Задание

По аналогии с примерами, приведенными в п. 6.1, 6.2, создать в БД XП, реализующие:

- вставку с пополнением справочников (например, вставляется информация о студенте, если указанный номер группы отсутствует в БД, запись добавляется в таблицу с перечнем групп);
- удаление с очисткой справочников (удаляется информация о студенте, если в его группе нет больше студентов, запись удаляется из таблицы с перечнем групп);
- каскадное удаление (при наличии условия ссылочной целостности по action перед удалением записи о группе удаляются записи обо всех студентах этой группы);
- вычисление и возврат значения агрегатной функции (на примере одного из запросов из задания);
- формирование статистики во временной таблице (например, для рассматриваемой БД — для каждого факультета: количество групп, количество обучающихся студентов, количество изучаемых дисциплин, средний балл по факультету).

Самостоятельно предложить и реализовать ПЗ или XП, демонстрирующие использование конструкций, описанных в п. 6.1.

Самостоятельно предложить и реализовать запросы, использующие скалярную и табличную функцию по аналогии с примерами в п. 6.3.

1.3 Содержание отчета

- текст запросов на SQL (с прояснениями/комментариями);
- наборы данных, возвращаемые запросами.

2 Выполнение работы

Исходные данные взяты из лабораторной работы №2, отчет для которой есть на GitHub (URI - https://github.com/vladcto/suai-labs/blob/d8c7a508971967641d8638ebcd107539c8fd618e/6_semester/%D0%9C%D0%A1%D0%9F%D0%98%D0%A1%D0%A2/%D0%BC%D1%81%D0%B8%D0%BF%D0%B8%D1%81%D1%82 2.pdf).

Исходный код доступен в приложении и в репозитории GitHub (URI - https://github.com/vladcto/suai-labs/tree/f380a9f5d145ba3ed6c0402c7dd570b98017a036/6_semester/%D0%9C%D0%A1%D0%9F%D0%98%D0%A1%D0%A2).

2.1 Вставка с пополнением справочников

Для вставки с пополнением справочника мы создаем хранимую процедуру InsertStudentWithGroupCheck, которая принимает в качестве входных параметров имя студента, название группы и ID факультета. Процедура выполняет проверку наличия указанной группы в базе данных. Если группа отсутствует, процедура добавляет новую запись в таблицу uni_group и присваивает ID последней добавленной записи переменной group_id. В случае, если группа уже присутствует в базе данных, процедура присваивает ID этой группы переменной group_id. Затем процедура добавляет новую запись в таблицу student с именем студента и ID группы.

Листинг insert_group.sql:

```
1 — ставляется информация о студенте, если указанный номер
      группы отсутствует
   -- в БД, запись добавляется в таблицу с перечнем групп
2
   USE conference db lab1;
3
4
5
   DELIMITER //
   CREATE PROCEDURE
6
7
      InsertStudentWithGroupCheck(IN student name VARCHAR(100),
8
                                   IN group name VARCHAR(50),
9
                                   IN faculty id int INT)
10
   BEGIN
11
     DECLARE group id INT;
12
13
     SELECT id INTO group id FROM uni group WHERE name =
         group name;
```

```
14
15
      IF group id IS NULL THEN
        INSERT INTO uni group (name, faculty id) VALUES (
16
           group name, faculty id int);
        SET group id = LAST INSERT ID();
17
      END IF;
18
19
      INSERT INTO student (name, group_id) VALUES (student_name,
20
         group id);
21
   END //
22
   DELIMITER ;
23
24
   CALL InsertStudentWithGroupCheck ('Новый', 'Новая', 1)
```

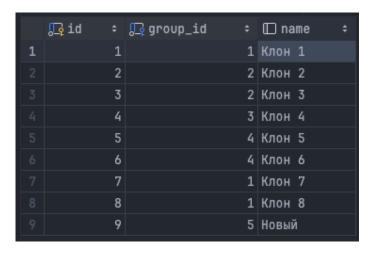


Рисунок 2.1 - Вставка студента

	<u>∏</u> id ÷	<u>∏</u> faculty_id :	÷	" name	
1	1		1	1	
2	2		2	2	
3	3		3	3	
4	4		4	4	
5	5		1	Новая	

Рисунок 2.2 - Вставка группы

2.2 Удаление с очисткой справочников

Для выполнения задания мы определяем хранимую процедуру DeleteStudentAndCleanGroup, которая принимает идентификатор студента в качестве входного параметра. Процедура извлекает идентификатор группы этого студента, затем удаляет запись студента из таблицы student.

После удаления студента процедура проверяет, остались ли в группе другие студенты. Если в группе не осталось студентов, процедура удаляет запись группы из таблицы uni group.

Листинг delete_student.sql:

```
-- удаляется информация о студенте, если в его группе нет
1
       больше студентов,
2
    -- удаляется из таблицы с перечнем групп
   USE conference db lab1;
3
4
5
   DELIMITER //
   CREATE PROCEDURE DeleteStudentAndCleanGroup(IN student id INT
       )
7
   BEGIN
8
      DECLARE s group id INT;
9
10
      SELECT group id INTO s group id FROM student WHERE id =
         student id;
11
12
      DELETE FROM student WHERE id = student_id;
13
14
      IF (SELECT COUNT(*) FROM student WHERE group id =
         s group id) = 0 THEN
15
       DELETE FROM uni group WHERE id = s group id;
16
     END IF;
   END//
17
18
   DELIMITER;
19
20
   CALL DeleteStudentAndCleanGroup (9);
```

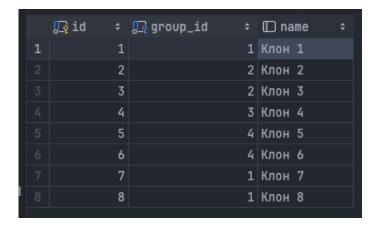


Рисунок 2.3 - Удаление студента

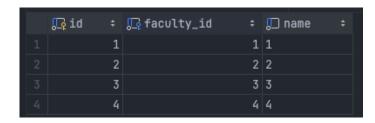


Рисунок 2.4 - Удаление группы

2.3 Каскадное удаление

Создадим XП(хранимую процедуру) DeleteFacultyCascade, которая принимает идентификатор факультета в качестве входного параметра. Процедура удаляет все записи студентов, связанных с группами, которые принадлежат указанному факультету. Затем процедура удаляет все записи групп, принадлежащих факультету. Наконец, процедура удаляет саму запись факультета.

Листинг faculity delete.sql:

```
1
   USE conference_db_lab1;
2
3
   DELIMITER //
4
   CREATE PROCEDURE DeleteFacultyCascade(IN faculty id INT)
5
   BEGIN
6
      DELETE student
7
       FROM student
8
               INNER JOIN uni group ON student.group id =
                  uni group.id
9
        WHERE uni group. faculty id = faculty id;
10
     DELETE FROM uni group WHERE uni group.faculty id =
11
         faculty id;
12
13
      DELETE FROM faculty WHERE id = faculty id;
   END//
14
   DELIMITER;
15
16
17
   CALL DeleteFacultyCascade(1);
```



Рисунок 2.5 - Удаление факультета



Рисунок 2.6 - Удаление группы

2.4 Вычисление и возврат значения агрегатной функции

Определим хранимую функцию (XФ) GetReportCountForUniversity, которая принимает идентификатор университета в качестве входного параметра и возвращает количество отчетов, созданных студентами этого университета. Функция выполняет запрос к таблицам authorship, student, uni_group и faculty, чтобы подсчитать количество отчетов, связанных с указанным университетом.

Листинг calculate student.sql:

```
-- Вычисление и возврат значения агрегатной функции
1
2
   USE conference db lab1;
3
4
   DELIMITER //
5
   CREATE FUNCTION GetReportCountForUniversity (university id INT
       ) RETURNS INT
   DETERMINISTIC
6
7
   BEGIN
8
     DECLARE report count INT;
9
10
     SELECT COUNT(*)
11
        INTO report count
12
       FROM authorship
13
               JOIN student ON authorship.author id = student.id
14
               JOIN uni group ON student.group_id = uni_group.id
15
               JOIN faculty ON uni_group.faculty_id = faculty.id
       WHERE faculty university id = university id;
16
17
```

```
RETURN report_count;

19 END//

20 DELIMITER;

21 SELECT GetReportCountForUniversity(1) as report_count;
```



Рисунок 2.7 - Количество докладов

2.5 Формирование статистики во временной таблице

SQL запрос определяет XП GenerateStatistics, которая формирует статистику по факультетам во временной таблице faculty_statistics. В этой таблице хранятся идентификаторы факультетов, количество групп и количество студентов для каждого факультета. Процедура выполняет запрос к таблицам faculty, uni_group и student, чтобы подсчитать количество групп и студентов для каждого факультета.

Листинг generate statistics.sql:

```
1
   -- формирование статистики во временной таблице
2
3
   USE conference db lab1;
4
5
   DELIMITER //
   CREATE PROCEDURE GenerateStatistics()
7
   BEGIN
8
      CREATE TEMPORARY TABLE IF NOT EXISTS faculty statistics
9
      (
10
        faculty id
                       INT,
                       INT,
11
        group_count
12
        student count INT
13
      );
14
      INSERT INTO faculty statistics (faculty_id, group_count,
15
         student count)
      SELECT f.id
16
                                   AS faculty id,
             COUNT(DISTINCT g.id) AS group_count,
17
```

```
18
             COUNT(DISTINCT s.id) AS student count
19
        FROM faculty f
20
               LEFT JOIN
21
                uni group g ON f.id = g.faculty id
22
               LEFT JOIN
23
                student s ON g.id = s.group id
24
        GROUP BY f.id;
25
   END//
26
   DELIMITER ;
27
28
   CALL GenerateStatistics();
    # noinspection SqlResolve
29
30
   SELECT *
31
      FROM faculty statistics;
```

∏ fac	culty_id ÷	☐ group_count ÷	□ student_count
1	2	1	2
2	3	1	1
3	4	1	2

Рисунок 2.8 - Получение статистики из временной таблицы

2.6 Запросы использующие скалярную и табличную функцию

В MySQL нельзя объявить функцию возвращающую тип таблицы. В MSSQL для этого можно было использовать синтаксис RETURNS TABLE AS RETURN ... в функции. В MySQL для этого можно использовать временную таблицу [1].

Поэтому SQL скрипт определяет функцию GetFacultyStudentCount и процедуру GetFacultyStatistics. Функция GetFacultyStudentCount принимает идентификатор факультета в качестве входного параметра и возвращает количество студентов, связанных с этим факультетом. Процедура GetFacultyStatistics создает временную таблицу FacultyStatistics, в которой хранятся идентификаторы факультетов, количество групп и количество студентов для каждого факультета.

Листинг faculty_statistics.sql:

```
1 USE conference_db_lab1;
2
```

```
3
   DELIMITER //
4
   CREATE FUNCTION GetFacultyStudentCount(faculty id INT)
      RETURNS INT
5
      DETERMINISTIC
6
   BEGIN
7
      DECLARE student count INT;
      SELECT COUNT(DISTINCT s.id)
8
9
        INTO student count
10
        FROM student s
11
               INNER JOIN uni group g ON s.group id = g.id
12
        WHERE g. faculty id = faculty id;
13
      RETURN student count;
14
   END//
15
   DELIMITER ;
16
17
   DELIMITER //
18
   CREATE PROCEDURE GetFacultyStatistics()
19
   BEGIN
      CREATE TEMPORARY TABLE IF NOT EXISTS Faculty Statistics AS
20
21
      SELECT f.id
                                            AS faculty_id,
22
             COUNT(DISTINCT g.id)
                                            AS group count,
             GetFacultyStudentCount(f.id) AS student_count
23
24
        FROM faculty f
25
               LEFT JOIN
26
               uni_group g ON f.id = g.faculty_id
27
        GROUP BY f.id;
28
29
      SELECT * FROM FacultyStatistics;
30
   END//
31
   DELIMITER ;
32
33
   CALL GetFacultyStatistics();
34
   # noinspection SqlResolve
35
   SELECT *
     FROM Faculty Statistics;
36
```

Рисунок 2.9 - Получение статистики

2.7 Запросы с операторами

Запрос определяет хранимую процедуру GetStudentsWithMoreThenNAuthorships, которая принимает количество авторств в качестве входного параметра. Процедура извлекает каждого студента из таблицы student и подсчитывает количество его авторств в таблице authorship. Если количество авторств студента превышает указанное значение, процедура выводит идентификатор и имя этого студента.

Листинг calculate topics.sql:

```
USE conference db lab1;
1
2
   DELIMITER //
3
   CREATE PROCEDURE GetStudentsWithMoreThenNAuthorships(IN
       n authorships INT)
5
   BEGIN
6
      DECLARE total students INT;
7
      DECLARE current row INT DEFAULT 1;
8
      DECLARE student id INT;
9
      DECLARE authorship count INT;
10
11
      SELECT COUNT(*) INTO total students FROM student;
12
13
      WHILE current row <= total students
14
        DO
15
          SELECT id INTO student id FROM student LIMIT
             current row, 1;
16
          SELECT COUNT(*)
17
            INTO authorship count
18
            FROM authorship
19
            WHERE author id = student id;
20
21
          IF authorship count > n authorships THEN
22
            SELECT id, name FROM student WHERE id = student id;
23
          END IF;
24
          SET current_row = current row + 1;
25
26
       END WHILE;
27
   END //
28
   DELIMITER;
```



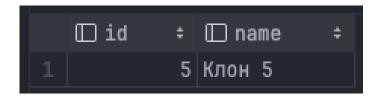


Рисунок 2.10 - Результат запроса с операторами

3 Вывод

В результате выполнения лабораторной работы были получены навыки работы с SQL-запросами, включая использование хранимых процедур и функций. Были учтены особенности их работы, а также показан результат выполнения этих команд.

Каждый запрос был разработан с учетом поставленных задач, а также внедрены самостоятельно предложенные запросы, демонстрирующие использование различных операторов SQL.

Полученные знания и навыки будут полезны в будущих проектах и задачах, связанных с обработкой данных в среде SQL.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. How to return table from MySQL function [Электронный pecypc]. URL: https://www.tutorialspoint.com/how-to-return-table-from-mysql-function (дата обращения: 25.05.2017).

ПРИЛОЖЕНИЕ

```
1
   insert group.sql
2
   -- ставляется информация о студенте, если указанный номер
      группы отсутствует
   — в БД, запись добавляется в таблицу с перечнем групп
3
4
   USE conference db lab1;
5
   DELIMITER //
6
7
   CREATE PROCEDURE
8
      InsertStudentWithGroupCheck(IN student name VARCHAR(100),
9
                                   IN group_name VARCHAR(50),
10
                                   IN faculty id int INT)
11
   BEGIN
12
      DECLARE group id INT;
13
14
     SELECT id INTO group id FROM uni group WHERE name =
         group name;
15
16
      IF group id IS NULL THEN
17
        INSERT INTO uni group (name, faculty id) VALUES (
           group_name , faculty_id_int);
18
        SET group_id = LAST_INSERT_ID();
19
      END IF:
20
21
      INSERT INTO student (name, group id) VALUES (student name,
         group id);
   END //
22
   DELIMITER :
23
24
    generate statistics.sql
   -- формирование статистики во временной таблице
25
26
27
   USE conference db lab1;
28
29
   DELIMITER //
30
   CREATE PROCEDURE Generate Statistics ()
31
   BEGIN
32
      CREATE TEMPORARY TABLE IF NOT EXISTS faculty_statistics
33
34
        faculty id
                      INT,
35
        group count
                      INT,
        student count INT
36
```

```
37
      );
38
39
      INSERT INTO faculty statistics (faculty id, group count,
         student count)
40
      SELECT f.id
                                   AS faculty id,
41
             COUNT(DISTINCT g.id) AS group count,
             COUNT(DISTINCT s.id) AS student count
42
       FROM faculty f
43
               LEFT JOIN
44
45
               uni group g ON f.id = g.faculty id
               LEFT JOIN
46
47
               student s ON g.id = s.group_id
       GROUP BY f.id;
48
49
   END//
50
   DELIMITER ;
51
52 CALL Generate Statistics ();
53
   # noinspection SqlResolve
54
   SELECT *
55
     FROM faculty_statistics;
    delete student.sql
56
57
   -- удаляется информация о студенте, если в его группе нет
       больше студентов,
    — удаляется из таблицы с перечнем групп
58
59
   USE conference db lab1;
60
61
   DELIMITER //
62
   CREATE PROCEDURE DeleteStudentAndCleanGroup(IN student id INT
   BEGIN
63
64
      DECLARE group id INT;
65
66
      SELECT group id INTO group id FROM student WHERE id =
         student id;
67
68
      DELETE FROM student WHERE id = student_id;
69
70
      IF (SELECT COUNT(*) FROM student WHERE group id = group id)
          = 0 THEN
71
       DELETE FROM uni_group WHERE id = group_id;
72
      END IF;
```

```
73
    END / /
74
    DELIMITER ;
75
76
    CALL DeleteStudentAndCleanGroup(1);
77
    faculty statistics.sql
78
    USE conference db lab1;
79
80
    DELIMITER //
81
    CREATE FUNCTION GetFacultyStudentCount(faculty id INT)
       RETURNS INT
82
       DETERMINISTIC
83
    BEGIN
84
      DECLARE student count INT;
85
      SELECT COUNT(DISTINCT s.id)
86
         INTO student count
87
        FROM student s
88
                INNER JOIN uni group g ON s.group id = g.id
89
        WHERE g. faculty id = faculty id;
90
      RETURN student count;
91
    END//
92
    DELIMITER ;
93
94
    DELIMITER //
95
    CREATE PROCEDURE GetFacultyStatistics()
96
97
      CREATE TEMPORARY TABLE IF NOT EXISTS Faculty Statistics AS
98
      SELECT f.id
                                             AS faculty id,
99
              COUNT(DISTINCT g.id)
                                             AS group count,
              GetFacultyStudentCount(f.id) AS student count
100
101
        FROM faculty f
102
                LEFT JOIN
103
                uni group g ON f.id = g.faculty id
104
        GROUP BY f.id;
105
106
      SELECT * FROM Faculty Statistics;
    END//
107
108
    DELIMITER ;
109
    CALL GetFacultyStatistics();
110
111
    # noinspection SqlResolve
    SELECT *
112
```

```
113
      FROM Faculty Statistics;
     faculity delete.sql
114
    USE conference db lab1;
115
116
117
    DELIMITER //
    CREATE PROCEDURE DeleteFacultyCascade(IN faculty id INT)
118
119
      DELETE student
120
        FROM student
121
122
                INNER JOIN uni group ON student.group id =
                   uni group.id
        WHERE uni group. faculty id = faculty id;
123
124
125
      DELETE FROM uni group WHERE uni group faculty id =
          faculty id;
126
127
      DELETE FROM faculty WHERE id = faculty id;
128
    END//
129
    DELIMITER ;
130
131
    CALL DeleteFacultyCascade(1);
     calculate topics.sql
132
133
    USE conference db lab1;
134
135
    DELIMITER //
136
    CREATE PROCEDURE GetStudentsWithMoreThenNAuthorships(IN
        n authorships INT)
137
    BEGIN
      DECLARE total students INT;
138
      DECLARE current row INT DEFAULT 1;
139
140
      DECLARE student id INT;
141
      DECLARE authorship count INT;
142
143
      SELECT COUNT(*) INTO total students FROM student;
144
145
       WHILE current row <= total students
146
        DO
147
           SELECT id INTO student id FROM student LIMIT
              current row, 1;
148
           SELECT COUNT(*)
             INTO authorship count
149
```

```
150
             FROM authorship
            WHERE author_id = student id;
151
152
153
           IF authorship count > n authorships THEN
             SELECT id, name FROM student WHERE id = student id;
154
155
          END IF;
156
157
           SET current row = current row + 1;
158
        END WHILE;
159
    END //
160
    DELIMITER ;
161
162
    CALL GetStudentsWithMoreThenNAuthorships (0);
163
    calculate student.sql
164
    — Вычисление и возврат значения агрегатной функции
    USE conference db lab1;
165
166
167
    DELIMITER //
168
    CREATE FUNCTION GetReportCountForUniversity (university id INT
       ) RETURNS INT
    BEGIN
169
170
      DECLARE report count INT;
171
172
      SELECT COUNT(*)
173
         INTO report count
174
        FROM authorship
175
                JOIN student ON authorship.author id = student.id
                JOIN uni group ON student.group id = uni group.id
176
177
                JOIN faculty ON uni group.faculty id = faculty.id
        WHERE faculty.university_id = university_id;
178
179
180
      RETURN report count;
181
    END//
182
    DELIMITER ;
183
184
    SELECT GetReportCountForUniversity(1);
```