

ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

Доцент				Бржезовский А. В.
должность, уч. степень, звание		подпись, дата		инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6

ХРАНИМЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ

Вариант 5

по курсу: Методы и средства проектирования информационных систем
и технологий

РАБОТУ ВЫПОЛНИЛ

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

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1	Постановка задачи	3
1.1	Цель работы	3
1.2	Задание	3
1.3	Содержание отчета	3
2	Выполнение 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	Вывод	14
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	15
	ПРИЛОЖЕНИЕ	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 ХП, 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 ХП, 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 пояснениями/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
2  -- в БД, запись добавляется в таблицу с перечнем групп
3  USE conference_db_lab1 ;
4
5  DELIMITER //
6  CREATE PROCEDURE
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
16     INSERT INTO uni_group (name, faculty_id) VALUES (
17         group_name, faculty_id_int);
18     SET group_id = LAST_INSERT_ID();
19 END IF;
20 INSERT INTO student (name, group_id) VALUES (student_name,
21     group_id);
22 END //
23 DELIMITER ;
24 CALL InsertStudentWithGroupCheck('Новый', 'Новая', 1)

```

	id	group_id	name
1	1	1	Клон 1
2	2	2	Клон 2
3	3	2	Клон 3
4	4	3	Клон 4
5	5	4	Клон 5
6	6	4	Клон 6
7	7	1	Клон 7
8	8	1	Клон 8
9	9	5	Новый

Рисунок 2.1 - Вставка студента

	id	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:

```
1  -- удаляется информация о студенте , если в его группе нет
    больше студентов ,
2  -- удаляется из таблицы с перечнем групп
3  USE conference_db_lab1 ;
4
5  DELIMITER //
6  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;
17 END//
18 DELIMITER ;
19
20 CALL DeleteStudentAndCleanGroup (9) ;
```

id	group_id	name
1	1	Клон 1
2	2	Клон 2
3	3	Клон 3
4	4	Клон 4
5	5	Клон 5
6	6	Клон 6
7	7	Клон 7
8	8	Клон 8

Рисунок 2.3 - Удаление студента

	id	faculty_id	name
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

Рисунок 2.4 - Удаление группы

2.3 Каскадное удаление

Создадим ХП(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 faculty_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
11     DELETE FROM uni_group WHERE uni_group.faculty_id =
        faculty_id ;
12
13     DELETE FROM faculty WHERE id = faculty_id ;
14 END//
15 DELIMITER ;
16
17 CALL DeleteFacultyCascade(1);

```

id	÷	university_id	÷	number	÷
1		2		1	2
2		3		1	3
3		4		2	4

Рисунок 2.5 - Удаление факультета

id	÷	faculty_id	÷	name	÷
1		2		2	2
2		3		3	3
3		4		4	4

Рисунок 2.6 - Удаление группы

2.4 Вычисление и возврат значения агрегатной функции

Определим хранимую функцию (ХФ) `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`:

```

1  -- Вычисление и возврат значения агрегатной функции
2  USE conference_db_lab1;
3
4  DELIMITER //
5  CREATE FUNCTION GetReportCountForUniversity(university_id INT
6      ) RETURNS INT
7  DETERMINISTIC
8  BEGIN
9
10     DECLARE report_count INT;
11
12     SELECT COUNT(*)
13     INTO report_count
14     FROM authorship
15     JOIN student ON authorship.author_id = student.id
16     JOIN uni_group ON student.group_id = uni_group.id
17     JOIN faculty ON uni_group.faculty_id = faculty.id
18     WHERE faculty.university_id = university_id;

```



```

18 RETURN report_count;
19 END//
20 DELIMITER ;
21
22 SELECT GetReportCountForUniversity(1) as report_count;

```



report_count	
1	3

Рисунок 2.7 - Количество докладов

2.5 Формирование статистики во временной таблице

SQL запрос определяет ХП `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 //
6  CREATE PROCEDURE GenerateStatistics()
7  BEGIN
8      CREATE TEMPORARY TABLE IF NOT EXISTS faculty_statistics
9      (
10         faculty_id    INT,
11         group_count   INT,
12         student_count INT
13     );
14
15     INSERT INTO faculty_statistics (faculty_id, group_count,
16                                     student_count)
17     SELECT f.id              AS faculty_id,
18            COUNT(DISTINCT g.id) AS group_count,

```

```

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();
29 # noinspection SqlResolve
30 SELECT *
31 FROM faculty_statistics;

```

	□ faculty_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)
5     RETURNS INT
6     DETERMINISTIC
7 BEGIN
8     DECLARE student_count INT;
9     SELECT COUNT(DISTINCT s.id)
10        INTO student_count
11        FROM student s
12             INNER JOIN uni_group g ON s.group_id = g.id
13             WHERE g.faculty_id = faculty_id;
14     RETURN student_count;
15 END//
16
17 DELIMITER //
18 CREATE PROCEDURE GetFacultyStatistics()
19 BEGIN
20     CREATE TEMPORARY TABLE IF NOT EXISTS FacultyStatistics AS
21     SELECT f.id                                AS faculty_id ,
22            COUNT(DISTINCT g.id)                AS group_count ,
23            GetFacultyStudentCount(f.id) AS student_count
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 *
36 FROM FacultyStatistics;

```

	□ faculty_id	÷	□ group_count	÷	□ student_count	÷
1		2		1		2
2		3		1		1
3		4		1		2

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`:

```
1  USE conference_db_lab1 ;
2
3  DELIMITER //
4  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
25         SET current_row = current_row + 1;
26     END WHILE;
27 END //
28 DELIMITER ;
```

29

30 `CALL GetStudentsWithMoreThanNAuthorships(0);`

	<input type="checkbox"/> id	÷	<input type="checkbox"/> name	÷
1		5	Клон 5	

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 ресурс]. 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
3  -- в БД, запись добавляется в таблицу с перечнем групп
4  USE conference_db_lab1;
5
6  DELIMITER //
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);
22  END //
23  DELIMITER ;
24  generate_statistics.sql
25  -- формирование статистики во временной таблице
26
27  USE conference_db_lab1;
28
29  DELIMITER //
30  CREATE PROCEDURE GenerateStatistics()
31  BEGIN
32      CREATE TEMPORARY TABLE IF NOT EXISTS faculty_statistics
33      (
34          faculty_id      INT,
35          group_count     INT,
36          student_count  INT
```



```

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 ,
42 COUNT(DISTINCT s.id) AS student_count
43 FROM faculty f
44 LEFT JOIN
45 uni_group g ON f.id = g.faculty_id
46 LEFT JOIN
47 student s ON g.id = s.group_id
48 GROUP BY f.id;
49 END//
50 DELIMITER ;
51
52 CALL GenerateStatistics();
53 # noinspection SqlResolve
54 SELECT *
55 FROM faculty_statistics;
56 delete_student.sql
57 -- удаляется информация о студенте, если в его группе нет
    больше студентов,
58 -- удаляется из таблицы с перечнем групп
59 USE conference_db_lab1;
60
61 DELIMITER //
62 CREATE PROCEDURE DeleteStudentAndCleanGroup(IN student_id INT
    )
63 BEGIN
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  BEGIN
97    CREATE TEMPORARY TABLE IF NOT EXISTS FacultyStatistics AS
98    SELECT f.id                                AS faculty_id ,
99          COUNT(DISTINCT g.id)                AS group_count ,
100         GetFacultyStudentCount(f.id) AS student_count
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 FacultyStatistics;
107  END//
108  DELIMITER ;
109
110  CALL GetFacultyStatistics();
111  # noinspection SqlResolve
112  SELECT *

```

```

113     FROM FacultyStatistics ;
114 faculty_delete.sql
115 USE conference_db_lab1 ;
116
117 DELIMITER //
118 CREATE PROCEDURE DeleteFacultyCascade(IN faculty_id INT)
119 BEGIN
120     DELETE student
121     FROM student
122         INNER JOIN uni_group ON student.group_id =
            uni_group.id
123     WHERE uni_group.faculty_id = faculty_id ;
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);
132 calculate_topics.sql
133 USE conference_db_lab1 ;
134
135 DELIMITER //
136 CREATE PROCEDURE GetStudentsWithMoreThanNAuthorships(IN
    n_authorships INT)
137 BEGIN
138     DECLARE total_students INT;
139     DECLARE current_row INT DEFAULT 1;
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(*)
149         INTO authorship_count

```

```

150         FROM authorship
151         WHERE author_id = student_id;
152
153     IF authorship_count > n_authorships THEN
154         SELECT id, name FROM student WHERE id = student_id;
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
165 USE conference_db_lab1;
166
167 DELIMITER //
168 CREATE FUNCTION GetReportCountForUniversity(university_id INT
169 ) RETURNS INT
170 BEGIN
171     DECLARE report_count INT;
172
173     SELECT COUNT(*)
174     INTO report_count
175     FROM authorship
176         JOIN student ON authorship.author_id = student.id
177         JOIN uni_group ON student.group_id = uni_group.id
178         JOIN faculty ON uni_group.faculty_id = faculty.id
179     WHERE faculty.university_id = university_id;
180
181     RETURN report_count;
182 END//
183
184 DELIMITER ;
185
186 SELECT GetReportCountForUniversity(1);

```