



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
МИРЭА – Российский технологический университет
РТУ МИРЭА
Колледж приборостроения и кибербезопасности

Практическая работа №9
по учебной дисциплине МДК.11.01
Технология разработки и защиты баз данных
специальность 09.02.07 Информационные системы и программирование
Создание триггеров

выполнил
студент группы ПКС-31
Лопатин Л.В.
преподаватель
Понеделко Е. В.

Москва

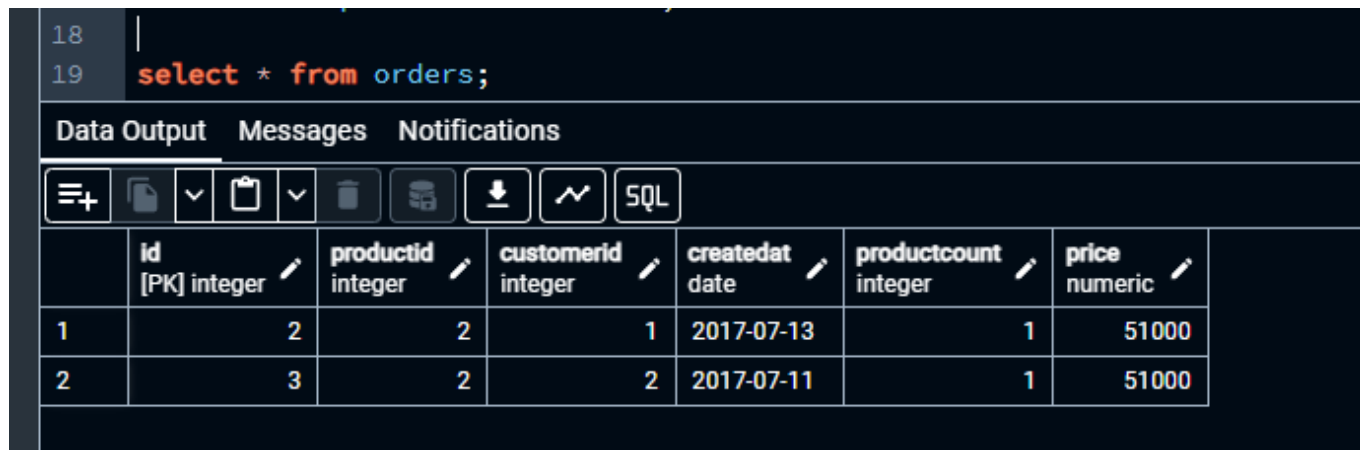
2025

Цель работы: Формирование навыков работы в среде СУБД PostgreSQL по созданию триггеров.

Ход работы.

Задание 1. Рассмотрел примеры и адаптировал код.

Пример 1. Для проверки работы триггера удалил запись из таблицы товаров(см. рисунок 1, 2)



The screenshot shows a PostgreSQL SQL client interface. At the top, a query is entered in a text area: `select * from orders;`. Below the query area, there are tabs for "Data Output", "Messages", and "Notifications". Under the "Data Output" tab, there is a toolbar with icons for various actions. Below the toolbar, a table of results is displayed. The table has 7 columns: "id", "productid", "customerid", "createdat", "productcount", and "price". The "id" column is marked as a primary key [PK] and integer. The "productid" column is marked as integer. The "customerid" column is marked as integer. The "createdat" column is marked as date. The "productcount" column is marked as integer. The "price" column is marked as numeric. There are two rows of data in the table.

	id [PK] integer	productid integer	customerid integer	createdat date	productcount integer	price numeric
1	2	2	1	2017-07-13	1	51000
2	3	2	2	2017-07-11	1	51000

Рис. 1 Таблица заказов до удаления

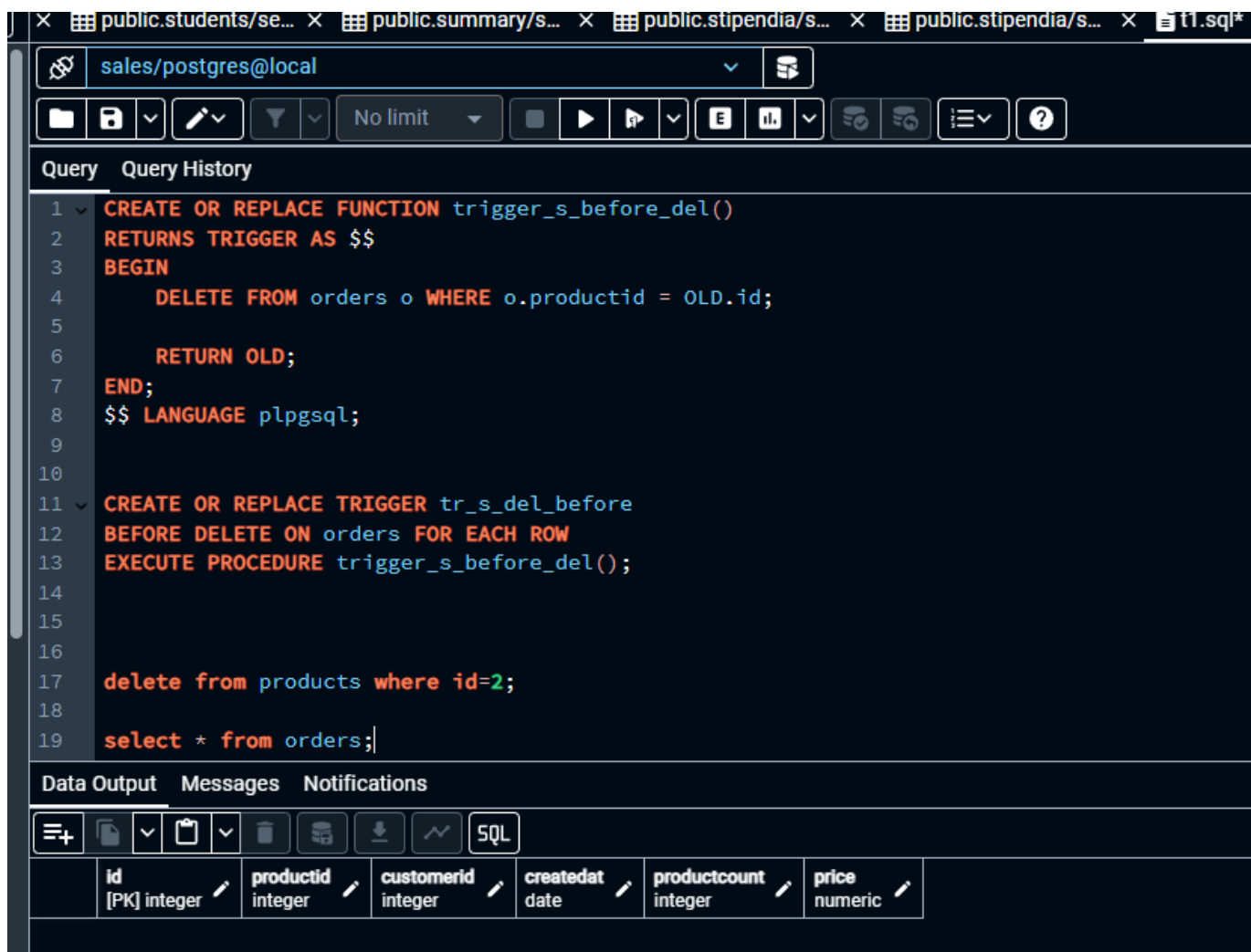


Рис. 2 Таблица заказов после удаления товара

Пример 2. Создал вспомогательную функцию для проверки корректности данных(см. рисунок 3). Создал триггерную функцию и триггер(см. рисунок 4). Результат представлен на рисунке 5.

```

CREATE FUNCTION iscorrect(integer, integer)
RETURNS BOOLEAN AS $$
BEGIN
-- проверка наличия студента
RETURN EXISTS(SELECT * FROM students WHERE id = $1)
-- проверка наличия плана, дисциплины и преподавателя
AND EXISTS(
    SELECT * FROM plan p
    INNER JOIN subject sub ON sub.id = p.idsubject
    INNER JOIN teachers t ON t.id = p.idteacher
    WHERE p.id = $2
);
END;
$$ LANGUAGE plpgsql;

```

Рис. 3 Вспомогательная функция

```

CREATE OR REPLACE FUNCTION trigger_results_insert()
RETURNS trigger AS $$
BEGIN
-- проверка корректности данных
IF iscorrect(NEW.idstudent, NEW.idplan) THEN
-- конвертация оценки(не используется из-за особенностей предметной области: 100-балльная оценка)
-- SELECT getmark3(NEW.grate) INTO NEW.grate;
SELECT NOW() INTO NEW.dateofexam;
ELSE
RAISE EXCEPTION 'Ошибка корректности данных';
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

Рис. 4 Триггерная функция

40

41

42

43

44

```

INSERT INTO summary(idstudent, idplan, grate) VALUES(13, 1, 77);

SELECT * FROM summary;

```

Data Output

Messages

Notifications

≡+

▼

▼

SQL

	dateofexam date	grate smallint	idstudent smallint	idplan smallint	id [PK] smallint
1	2025-02-20	100	13	1	26
2	2025-02-23	30	12	1	27
3	2025-02-23	100	11	1	28
4	2025-02-23	100	11	1	29
5	2025-02-23	100	11	1	30
6	2025-02-23	100	13	1	31
7	2025-02-23	100	13	1	32
8	2025-02-23	77	13	1	33
9	2025-02-20	80	13	1	16
10	2025-02-20	80	13	1	17

Рис. 5 Проверка работы

Пример 3. Создал таблицу rating. Создал триггерную функцию и триггер(см. рисунок 6). Результат представлен на рисунке 7.

Query Query History

```

1 CREATE FUNCTION trigger_results_insert_after()
2 RETURNS trigger AS
3 $$ BEGIN
4     IF EXISTS(SELECT * FROM rating WHERE
5 idstudent=NEW.idstudent) THEN UPDATE rating SET
6 summ_grate=summ_grate+NEW.grate
7 WHERE idstudent=NEW.idstudent;
8 ELSE
9 INSERT INTO rating (idstudent, summ_grate)
10 VALUES (NEW.idstudent, NEW.grate);
11 END IF;
12 RETURN NEW;
13 END;
14 $$
15 LANGUAGE plpgsql;
16 CREATE TRIGGER tr_results_insert_after AFTER INSERT ON
17 summary FOR EACH ROW
18 EXECUTE PROCEDURE trigger_results_insert_after();

```

Рис. 6 Триггерная функция

The screenshot shows a SQL IDE with a dark theme. The top panel contains SQL code for a trigger function. The bottom panel shows the 'Data Output' tab with a table of results.

```

19
20 INSERT INTO summary(grate, idstudent, idplan) VALUES (100, 11, 1);
21
22 SELECT * FROM rating;
  
```

Below the code editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with two columns: 'idstudent' (labeled '[PK] smallint') and 'summ_grate' (labeled 'integer'). The table contains two rows of data.

	idstudent [PK] smallint	summ_grate integer
1	11	200
2	13	277

Рис. 7 Обновлённый рейтинг

Задание 2. Проанализировал базу данных “Сессия” и создал 3 триггера.

Триггер BEFORE. Создал триггер для предотвращения добавления некорректных записей в план. Результат представлен на рисунке 8.

The screenshot shows a SQL IDE with a dark theme. The top panel contains SQL code for a trigger function. The bottom panel shows the 'Data Output' tab with an error message.

```

1 -- Необходимо предотвращать попытки формирования плана для несуществующих предметов и преподавателей
2
3 DROP TRIGGER tr_before ON students;
4 DROP FUNCTION trigger_before;
5
6 CREATE OR REPLACE FUNCTION trigger_before()
7 RETURNS trigger AS $$
8 BEGIN
9 IF EXISTS(SELECT id FROM teachers t WHERE t.id = NEW.idteacher) AND
10 EXISTS(SELECT id FROM subject sub WHERE sub.id = NEW.idsubject) THEN
11 ELSE
12 RAISE EXCEPTION 'Ошибка корректности данных';
13 END IF;
14 RETURN NEW;
15 END;
16 $$ LANGUAGE plpgsql;
17
18 CREATE TRIGGER tr_before
19 BEFORE INSERT ON plan FOR EACH ROW
20 EXECUTE FUNCTION trigger_before();
21
22 INSERT INTO plan(semester, examform, hours, idteacher, idsubject) VALUES(3, 'Экзамен', 100, 100, 1);
  
```

Below the code editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing an error message.

ERROR: Ошибка корректности данных
 CONTEXT: PL/pgSQL function trigger_before() line 6 at RAISE
 SQL state: P0001

Рис. 8 Попытка ввода некорректных данных

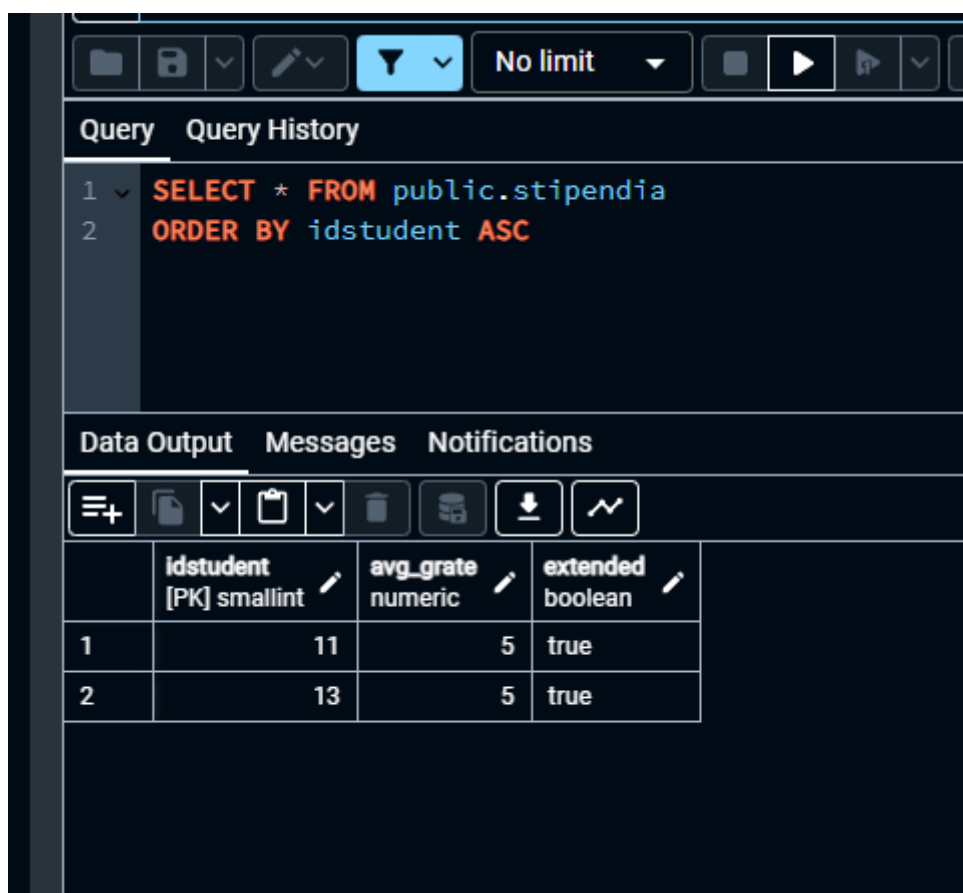
Триггер AFTER. Создал триггер для обновления списка стипендиатов при добавлении оценок в ведомость(см. рисунок 9). Результат представлен на рисунке 10.

```

Query  Query History
1  -- Необходимо обновлять список стипендиатов каждый раз при изменении сводной ведомости
2
3  DROP TRIGGER tr_after ON summary;
4  DROP FUNCTION trigger_after;
5
6  CREATE OR REPLACE FUNCTION trigger_after()
7  RETURNS TRIGGER AS $$
8  BEGIN
9      PERFORM updatestipendia(1);
10
11     RETURN NEW;
12 END;
13 $$ LANGUAGE plpgsql;
14
15 CREATE TRIGGER tr_after
16 AFTER INSERT ON summary
17 FOR EACH ROW
18 EXECUTE FUNCTION trigger_after();
19
20
21 INSERT INTO summary(idstudent, idplan, grate) VALUES(13,1,100)

```

Рис. 9 Триггер обновления списка стипендиатов



The screenshot shows a database client interface with a toolbar at the top containing icons for file operations, a filter icon, and a 'No limit' dropdown. Below the toolbar are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query:

```

1 SELECT * FROM public.stipendia
2 ORDER BY idstudent ASC

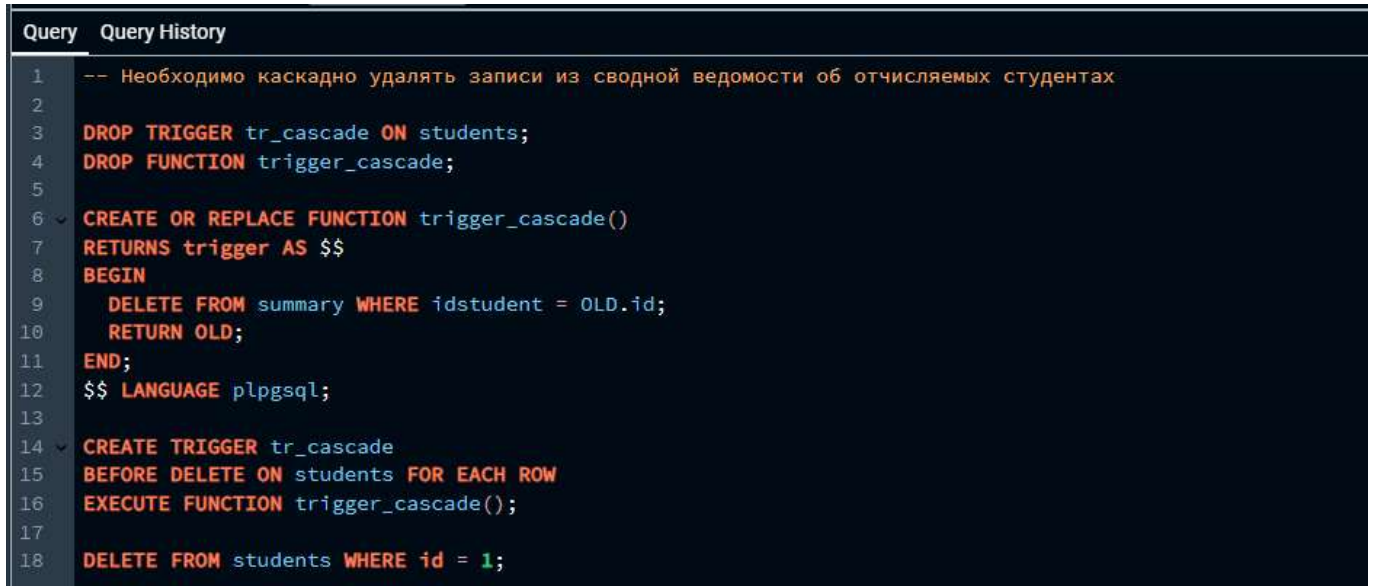
```

Below the query editor are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the results of the query. The table has four columns: 'idstudent' (smallint, PK), 'avg_grate' (numeric), and 'extended' (boolean). There are two rows of data.

	idstudent [PK] smallint	avg_grate numeric	extended boolean
1	11	5	true
2	13	5	true

Рис. 10 Обновлённый список стипендиатов

Триггер каскадного удаления Создал триггер для удаления записей из ведомости об отчисленных студентах(см. рисунок 11). Результат представлен на рисунке 12.



```
Query  Query History
1  -- Необходимо каскадно удалять записи из сводной ведомости об отчисляемых студентах
2
3  DROP TRIGGER tr_cascade ON students;
4  DROP FUNCTION trigger_cascade;
5
6  CREATE OR REPLACE FUNCTION trigger_cascade()
7  RETURNS trigger AS $$
8  BEGIN
9      DELETE FROM summary WHERE idstudent = OLD.id;
10     RETURN OLD;
11 END;
12 $$ LANGUAGE plpgsql;
13
14 CREATE TRIGGER tr_cascade
15 BEFORE DELETE ON students FOR EACH ROW
16 EXECUTE FUNCTION trigger_cascade();
17
18 DELETE FROM students WHERE id = 1;
```

Рис. 11 Триггер каскадного удаления

