|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра практической и прикладной информатики (ППИ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Разработка баз данных»

**Практическое занятие № 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | ИКБО-20-21 Хитров Н.С. | (подпись) | |
| Преподаватель | Благовещенский В.Г. | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_2023 г. | |  | |

Москва 2023 г.

**Цель занятия:**

Написание триггеров, функций и процедур различной степени сложности.

**Постановка задачи:**

Изучить и создать несколько (по 3) триггеров, функций и процедур.

Результат работы в виде отчета должен содержать:

- снимок экрана (скриншоты) триггеров, функций и процедур

- снимок экрана (скриншоты) результата работы триггеров, функций и процедур;

- снимки экрана (скриншоты)

**Результат работы:**

**Написание триггеров**

Триггер, который обновляет статус резюме при добавлении фидбэка в соответствие с заключением, переданным в фидбэк.

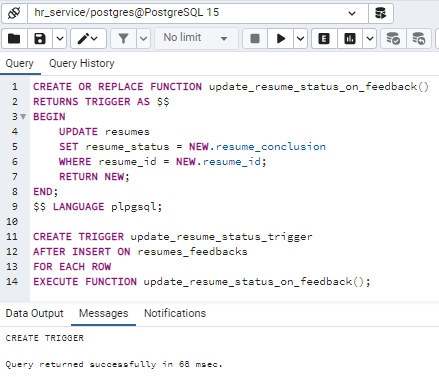


Рисунок 1 – Триггер update\_resume\_status\_on\_feedback

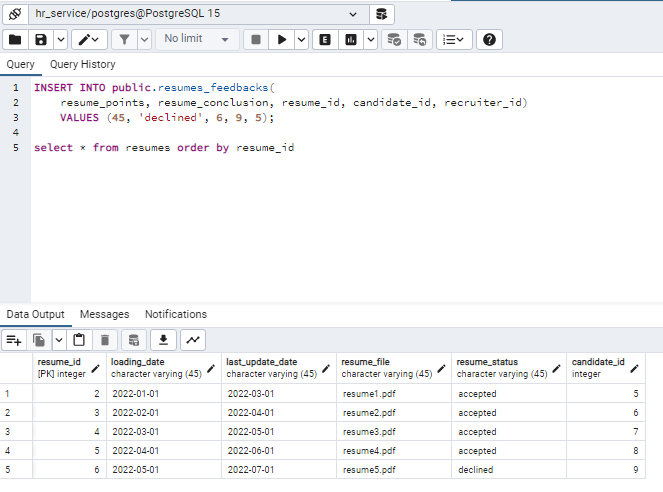


Рисунок 2 – Результат работы триггера

Триггер, который присваивает кандидату случайный набор заданий для собеседования при условии принятия его резюме.

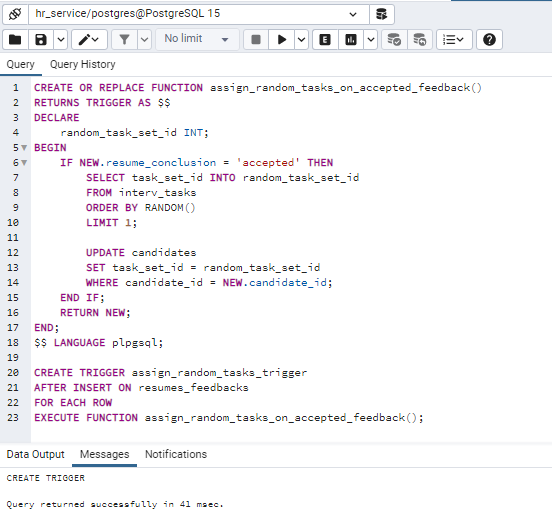


Рисунок 7 – Триггер assign\_random\_tasks\_on\_accepted\_feedback

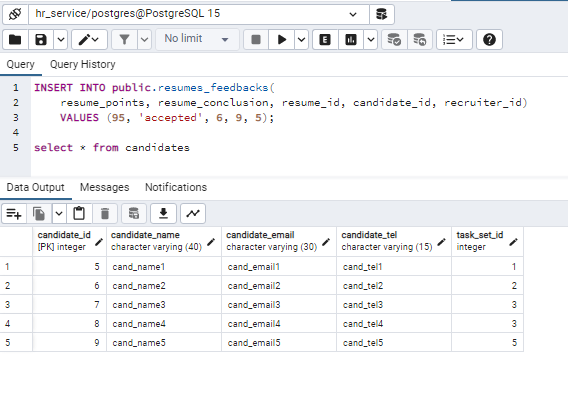


Рисунок 3 – Результат работы триггера

Триггер, который при обновлении столбца task\_set\_id в записи таблицы кандидатов просчитывает и обновляет столбец кол-ва курируемых соответствующим специалистом кандидатов.

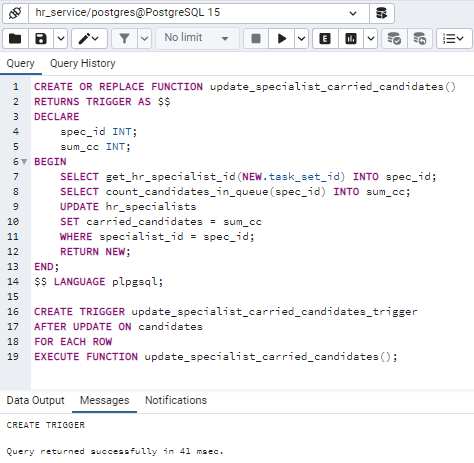


Рисунок 4 – Триггер update\_specialist\_carried\_candidates

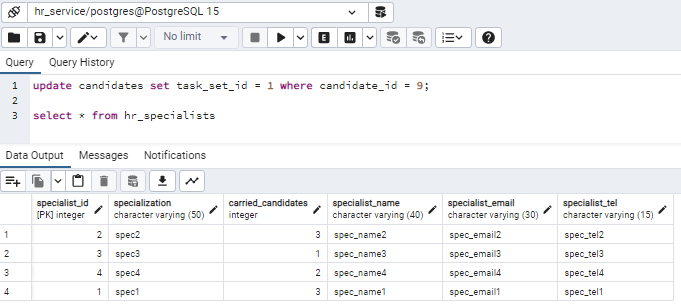


Рисунок 5 – Результат работы триггера

**Написание функций**

Функция, которая рассчитывает кол-во кандидатов курируемых соответствующим специалистом.



Рисунок 6 – Функция count\_candidates\_in\_queue

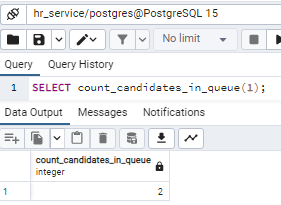


Рисунок 7 – Вызов функции

Функция, которая возвращает id специалиста по id набора заданий который он курирует;

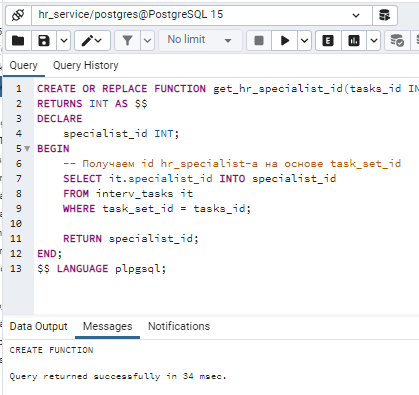


Рисунок 8 – Функция get\_hr\_specialist\_id

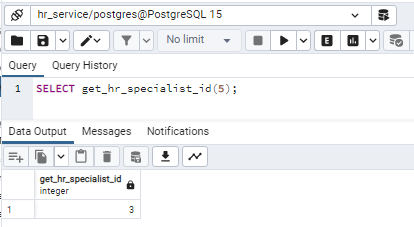


Рисунок 9 – Вызов функции

**Написание процедур**

Процедура, которая удаляет работника с должности интервьюера.

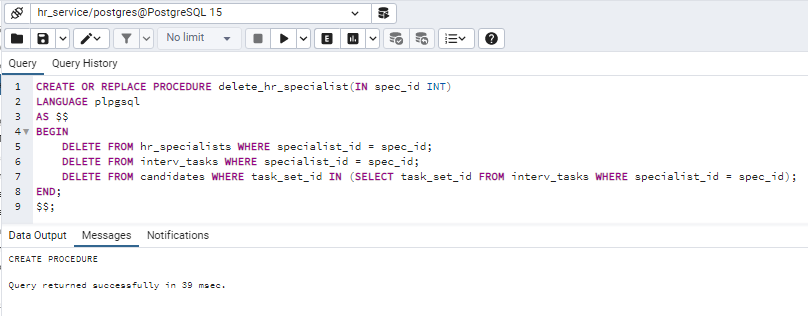


Рисунок 12 – Процедура delete\_hr\_specialist

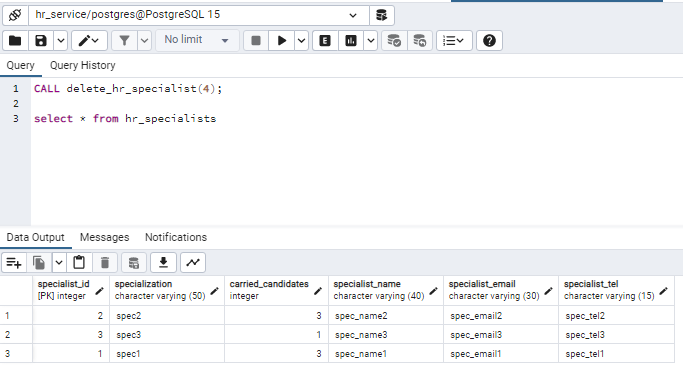


Рисунок 13 – Результат работы процедуры

Процедура, которая добавляет фидбэк на резюме.

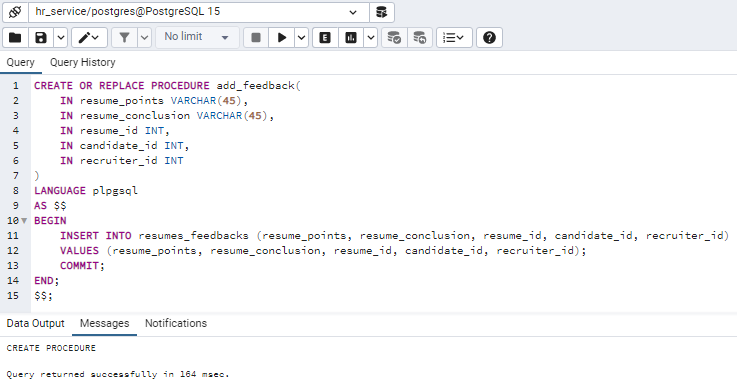


Рисунок 14 – Процедура add\_feedback

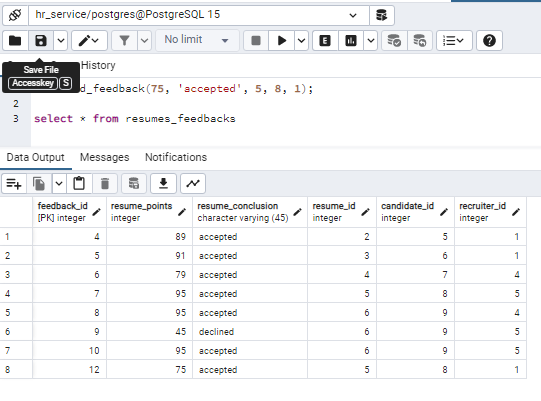


Рисунок 14 – результат вызова процедуры

Процедура, которая создает резюме пользователя.

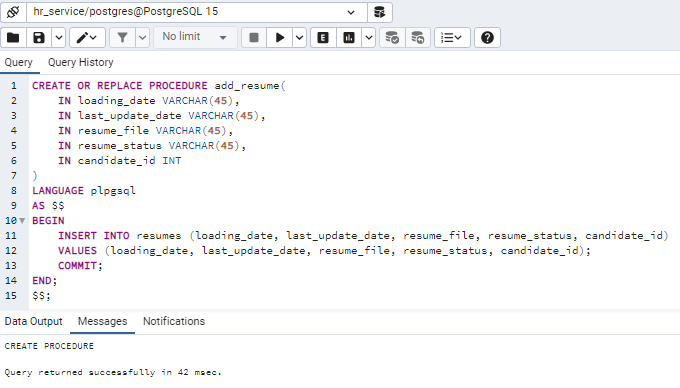


Рисунок 15 – Процедура add\_resume

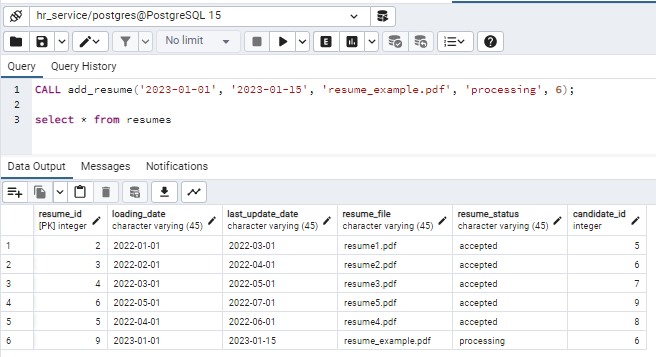


Рисунок 16 – Результат вызова процедуры

Для переноса базы данных использовался функционал MySQL Workbench – функция Export data. Результат представлен на рисунке 25.

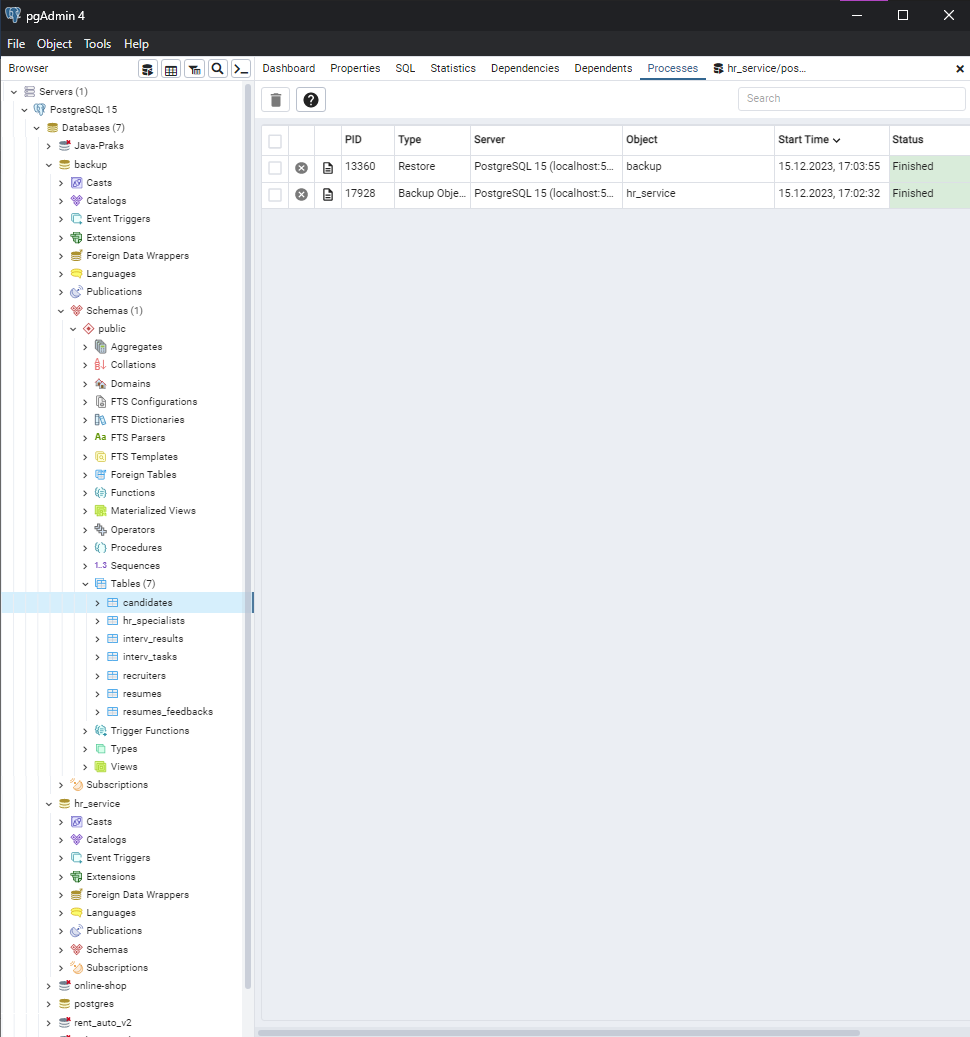


Рисунок 17 – Результат экспорта БД

Листинг 1 – SQL код созданных функций, триггеров и процедур, а также их проверок

|  |
| --- |
| CREATE OR REPLACE FUNCTION count\_candidates\_in\_queue(interviewer\_id INT)  RETURNS INTEGER AS $$  DECLARE  queue\_count INTEGER;  BEGIN  SELECT COUNT(\*) INTO queue\_count  FROM candidates  WHERE task\_set\_id IN (SELECT task\_set\_id FROM interv\_tasks WHERE specialist\_id = interviewer\_id);  RETURN queue\_count;  END;  $$ LANGUAGE plpgsql;  select \* from hr\_specialists  select \* from candidates  select \* from interv\_tasks  SELECT count\_candidates\_in\_queue(1);  CREATE OR REPLACE FUNCTION get\_hr\_specialist\_id(tasks\_id INT)  RETURNS INT AS $$  DECLARE  specialist\_id INT;  BEGIN  SELECT it.specialist\_id INTO specialist\_id  FROM interv\_tasks it  WHERE task\_set\_id = tasks\_id;  RETURN specialist\_id;  END;  $$ LANGUAGE plpgsql;  select \* from hr\_specialists  select \* from candidates  select \* from interv\_tasks  SELECT get\_hr\_specialist\_id(5);  CREATE OR REPLACE FUNCTION update\_resume\_status\_on\_feedback()  RETURNS TRIGGER AS $$  BEGIN  UPDATE resumes  SET resume\_status = NEW.resume\_conclusion  WHERE resume\_id = NEW.resume\_id;  RETURN NEW;  END;  $$ LANGUAGE plpgsql;  CREATE TRIGGER update\_resume\_status\_trigger  AFTER INSERT ON resumes\_feedbacks  FOR EACH ROW  EXECUTE FUNCTION update\_resume\_status\_on\_feedback();  select \* from resumes order by resume  INSERT INTO public.resumes\_feedbacks(  resume\_points, resume\_conclusion, resume\_id, candidate\_id, recruiter\_id)  VALUES (45, 'declined', 6, 9, 5);    CREATE OR REPLACE FUNCTION assign\_random\_tasks\_on\_accepted\_feedback()  RETURNS TRIGGER AS $$  DECLARE  random\_task\_set\_id INT;  BEGIN  IF NEW.resume\_conclusion = 'accepted' THEN  SELECT task\_set\_id INTO random\_task\_set\_id  FROM interv\_tasks  ORDER BY RANDOM()  LIMIT 1;  UPDATE candidates  SET task\_set\_id = random\_task\_set\_id  WHERE candidate\_id = NEW.candidate\_id;  END IF;  RETURN NEW;  END;  $$ LANGUAGE plpgsql;  CREATE TRIGGER assign\_random\_tasks\_trigger  AFTER INSERT ON resumes\_feedbacks  FOR EACH ROW  EXECUTE FUNCTION assign\_random\_tasks\_on\_accepted\_feedback();  INSERT INTO public.resumes\_feedbacks(  resume\_points, resume\_conclusion, resume\_id, candidate\_id, recruiter\_id)  VALUES (95, 'accepted', 6, 9, 5);  select \* from candidates      CREATE OR REPLACE FUNCTION update\_specialist\_carried\_candidates()  RETURNS TRIGGER AS $$  DECLARE  spec\_id INT;  sum\_cc INT;  BEGIN  SELECT get\_hr\_specialist\_id(NEW.task\_set\_id) INTO spec\_id;  SELECT count\_candidates\_in\_queue(spec\_id) INTO sum\_cc;  UPDATE hr\_specialists  SET carried\_candidates = sum\_cc  WHERE specialist\_id = spec\_id;  RETURN NEW;  END;  $$ LANGUAGE plpgsql;  CREATE TRIGGER update\_specialist\_carried\_candidates\_trigger  AFTER UPDATE ON candidates  FOR EACH ROW  EXECUTE FUNCTION update\_specialist\_carried\_candidates();  update candidates set task\_set\_id = 1 where candidate\_id = 33  select \* from hr\_specialists  CREATE OR REPLACE PROCEDURE delete\_hr\_specialist(IN spec\_id INT)  LANGUAGE plpgsql  AS $$  BEGIN  DELETE FROM hr\_specialists WHERE specialist\_id = spec\_id;  DELETE FROM interv\_tasks WHERE specialist\_id = spec\_id;  DELETE FROM candidates WHERE task\_set\_id IN (SELECT task\_set\_id FROM interv\_tasks WHERE specialist\_id = spec\_id);  END;  $$;  CALL delete\_hr\_specialist(4);  CREATE OR REPLACE PROCEDURE add\_feedback(  IN resume\_points INT,  IN resume\_conclusion VARCHAR(45),  IN resume\_id INT,  IN candidate\_id INT,  IN recruiter\_id INT  )  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO resumes\_feedbacks (resume\_points, resume\_conclusion, resume\_id, candidate\_id, recruiter\_id)  VALUES (resume\_points, resume\_conclusion, resume\_id, candidate\_id, recruiter\_id);  COMMIT;  END;  $$;  CALL add\_feedback(75, 'аccepted', 5, 8, 1);  select \* from resumes\_feedbacks  CREATE OR REPLACE PROCEDURE add\_resume(  IN loading\_date VARCHAR(45),  IN last\_update\_date VARCHAR(45),  IN resume\_file VARCHAR(45),  IN resume\_status VARCHAR(45),  IN candidate\_id INT  )  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO resumes (loading\_date, last\_update\_date, resume\_file, resume\_status, candidate\_id)  VALUES (loading\_date, last\_update\_date, resume\_file, resume\_status, candidate\_id);  COMMIT;  END;  $$;  CALL add\_resume('2023-01-01', '2023-01-15', 'resume\_example.pdf', 'processing', 6);  select \* from resumes |

**Вывод:**

В результате данной практической работы были созданы три процедуры, три функции и три триггера. А также создан backup для базы данных.

**Список использованных источников и литературы:**

1. Методические указания к практической работе по дисциплине “Разработка баз данных” [Электронный ресурс]: Режим доступа: <https://online-edu.mirea.ru/mod/folder/view.php?id=263588> [Дата обращения : 13.09.2023]