# Отчёт по работе №3

## Представления, хр. процедуры, триггера и курсоры.

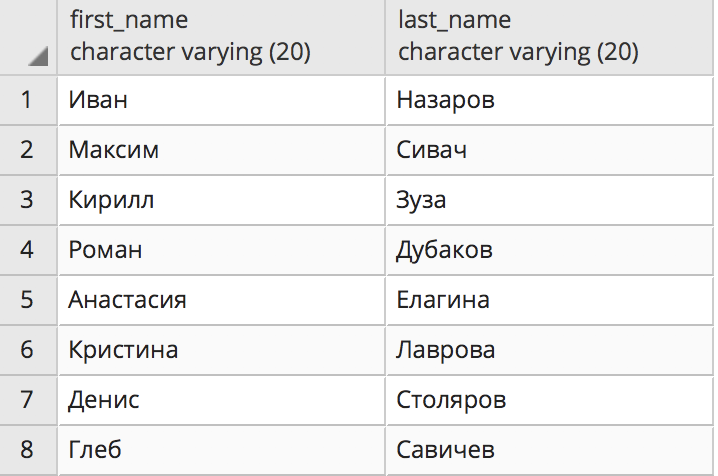
### Представления

1. Создать представление, отображающее всех водителей

CREATE OR REPLACE VIEW first\_v

AS SELECT first\_name, last\_name FROM auto\_personnel;

Результат:



1. Создать представление, отображающее все маршруты и количество автомобилей, находящихся на каждом маршруте

CREATE OR REPLACE VIEW second\_v

AS SELECT routes.name, Count(journal.auto\_id)

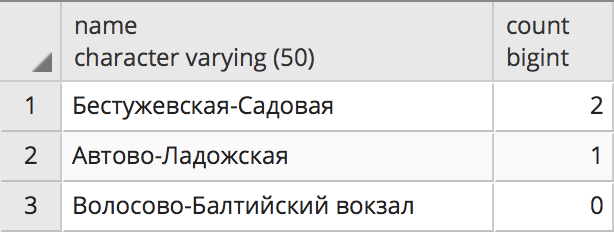
FROM routes FULL JOIN journal

ON routes.id = journal.route\_id

WHERE journal.time\_in IS NULL

GROUP BY routes.name;

Результат:



Hints: select, where, count, max, group by, having, like, create view, drop view

### Хранимые процедуры

* без параметров

1. Создать хранимую процедуру, выводящую все маршруты и среднее время проезда по ним в минутах

CREATE OR REPLACE FUNCTION proc\_first ()

returns table(Маршрут varchar(50), Среднее\_время\_в\_минутах double precision) AS

$body$

begin

return query

SELECT routes.name, avg(extract(epoch from (journal.time\_in - journal.time\_out)))/60

FROM routes, journal

WHERE routes.id = journal.route\_id

GROUP BY routes.name;

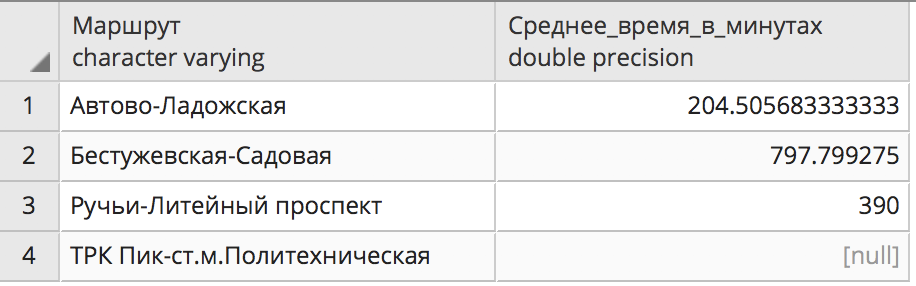
end;

$body$

LANGUAGE 'plpgsql';

select \* from proc\_first();

Результаты:



* с входными параметрами

1. Создать хранимую процедуру, имеющую два параметра автомобиль1 и автомобиль2. Она должна возвращать маршруты, по которым автомобиль1 проехал быстрее автомобиля2. Если на каком-либо маршруте один из автомобилей не проезжал, такой маршрут не рассматривается.

CREATE OR REPLACE FUNCTION proc\_second (auto1 varchar(50), auto2 varchar(50) )

returns table(Routes varchar(50)) AS

$body$

begin

return query SELECT name FROM routes WHERE id = (SELECT route FROM

(SELECT \* FROM

(SELECT journal.route\_id AS route, auto\_id, journal.time\_in - journal.time\_out AS col FROM journal

WHERE

journal.auto\_id = (SELECT id FROM auto WHERE num = auto1)

OR

journal.auto\_id = (SELECT id FROM auto WHERE num = auto2)) AS t1

WHERE (auto\_id,col) IN

( SELECT auto\_id, MAX(col) FROM

(SELECT journal.route\_id, auto\_id, journal.time\_in - journal.time\_out AS col FROM journal

WHERE

journal.auto\_id = (SELECT id FROM auto WHERE num = auto1)

OR

journal.auto\_id = (SELECT id FROM auto WHERE num = auto2)) AS t1

GROUP BY auto\_id

)

)AS t2

WHERE auto\_id = (SELECT id FROM auto WHERE num = auto1));

end;

$body$

LANGUAGE 'plpgsql';

SELECT \* FROM proc\_second('P632MM', 'K772MA');

Результаты:

Таблица routes:

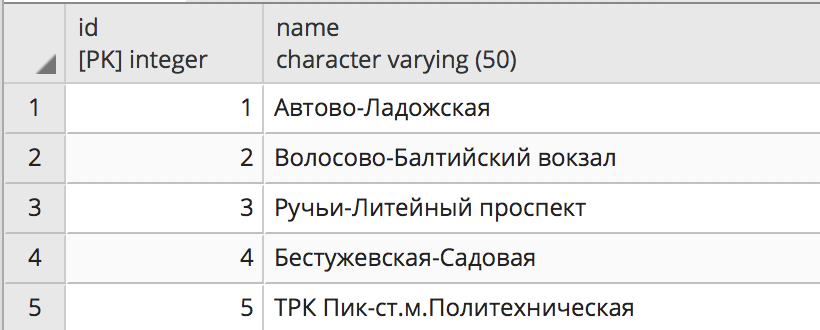
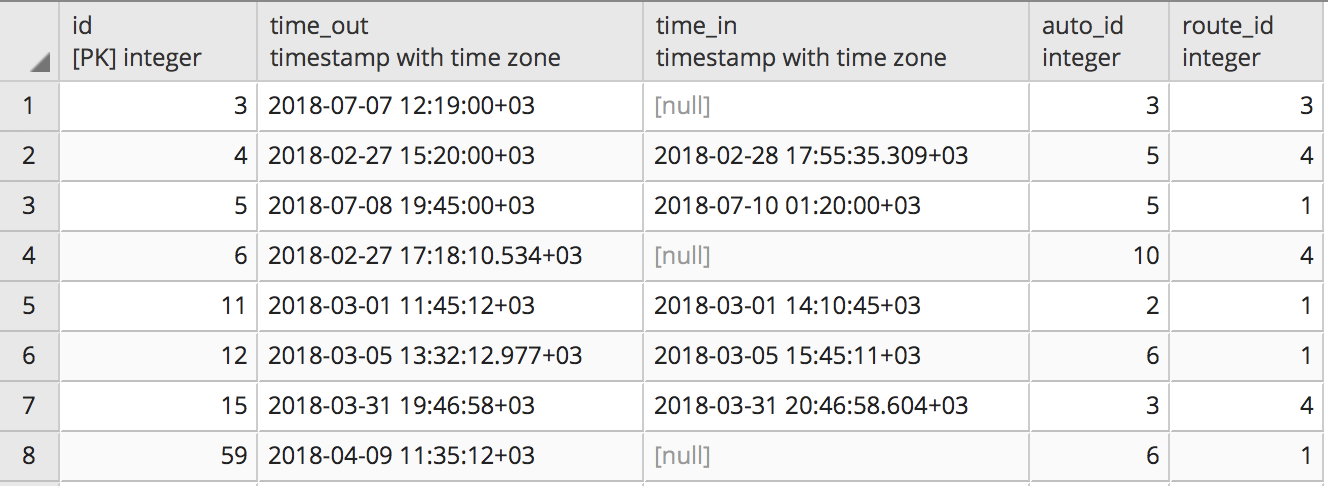


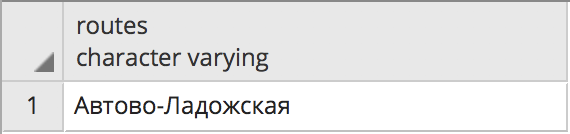
Таблица auto:



Таблица journal:



Получение маршрута, по которому автомобиль с номером «P632MM» проехал быстрее автомобиля с номером «K772MA» (в результате должен быть маршрут с id = 1):



* с выходными параметрами

1. Создать хранимую процедуру с входным параметром маршрут и двумя выходными параметрами, возвращающими самое короткое время проезда по заданному маршруту и автомобиль, поставивший рекорд

CREATE OR REPLACE FUNCTION proc\_third (name\_route varchar(50))

returns table(Самое\_короткое\_время\_по\_маршруту interval, Автомобиль varchar(20)) AS

$body$

begin

return query SELECT MIN(journal.time\_in - journal.time\_out), auto.num

FROM journal, routes, auto

WHERE (routes.id = journal.route\_id) AND (journal.auto\_id = auto.id) AND (routes.name = name\_route)

GROUP BY journal.time\_in - journal.time\_out, auto.num;

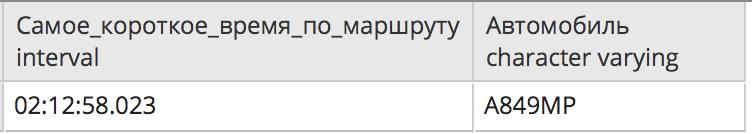
end;

$body$

LANGUAGE 'plpgsql';

SELECT \* FROM proc\_third('Автово-Ладожская');

Результат:



Hints: select, where, count, max, group by, having, create procedure, drop procedure

### Триггера

* Триггера на вставку

1. Создать триггер, который не позволяет добавить автомобиль с номером, который уже есть в парке

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION insert\_auto\_by\_num () RETURNS trigger AS $first\_insert$

BEGIN

if (SELECT Count(\*) FROM auto WHERE NEW.num = num) = 0

then

return NEW;

else

RAISE 'Уже есть автомобиль с таким номером';

return OLD;

end if;

END;

$first\_insert$ LANGUAGE plpgsql;

CREATE TRIGGER first\_insert

BEFORE INSERT

ON auto

for each row

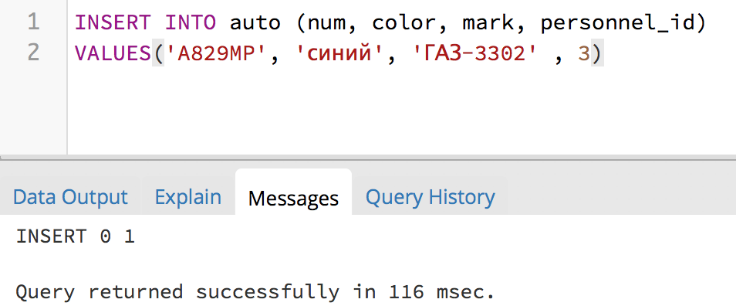
EXECUTE PROCEDURE insert\_auto\_by\_num();

Результаты:

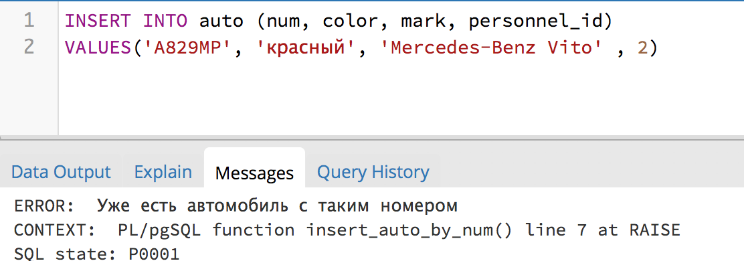
Таблица auto:



Добавление автомобиля с несуществующим номером:



Добавление автомобиля с таким же номером:



1. Создать триггер, который не позволяет добавить запись в журнал оператора, в которой дата прибытия не NULL

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION no\_insert\_entry\_without\_null() RETURNS trigger AS $second\_insert$

BEGIN

if NEW.time\_in is NULL then

RAISE 'Запись успешно добавлена!';

return NEW;

else

RAISE 'Дата прибытия должна быть NULL!';

return OLD;

end if;

END;

$second\_insert$ LANGUAGE plpgsql;

CREATE TRIGGER second\_insert

BEFORE INSERT

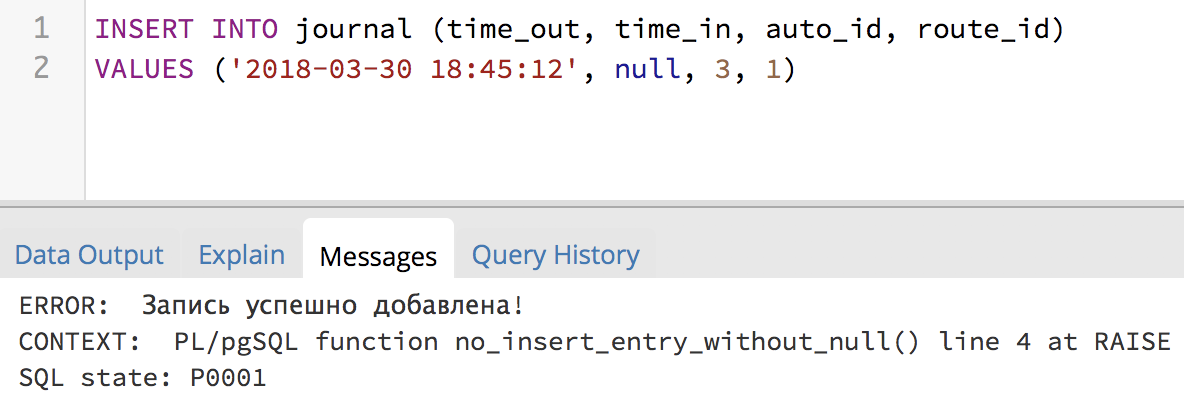
ON journal

for each row

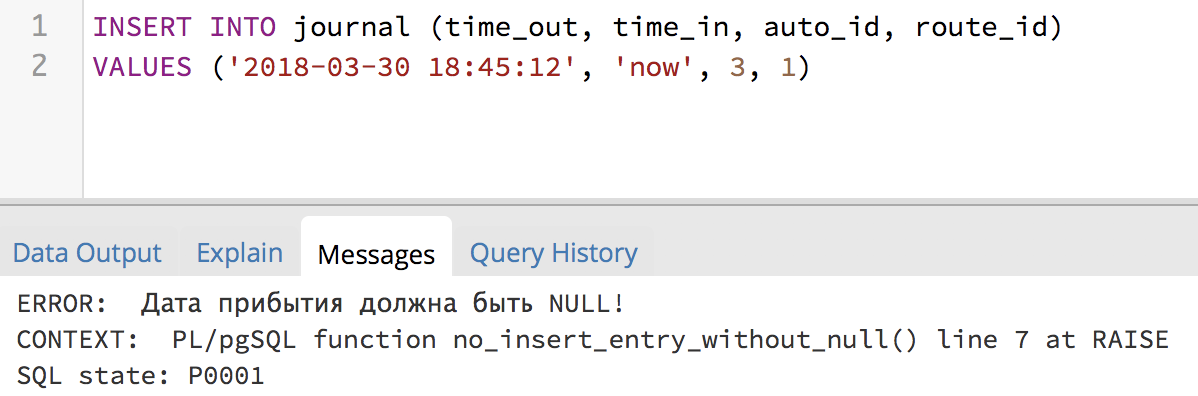
EXECUTE PROCEDURE no\_insert\_entry\_without\_null();

Результаты:

Добавление записи в журнал оператора, в которой дата прибытия NULL:



Добавление записи в журнал оператора, в которой дата прибытия не NULL:



1. Создать триггер, который не позволяет отправить в рейс автомобиль, который еще не вернулся в парк

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION no\_insert\_auto\_without\_time\_in() RETURNS trigger AS $second\_insert$

BEGIN

if (SELECT Count(\*) FROM journal WHERE (auto\_id = NEW.auto\_id) AND time\_in is NULL) = 0 then

return NEW;

else

RAISE 'Этот автомобиль ещё не вернулся!';

return OLD;

end if;

END;

$second\_insert$ LANGUAGE plpgsql;

CREATE TRIGGER third\_insert

BEFORE INSERT

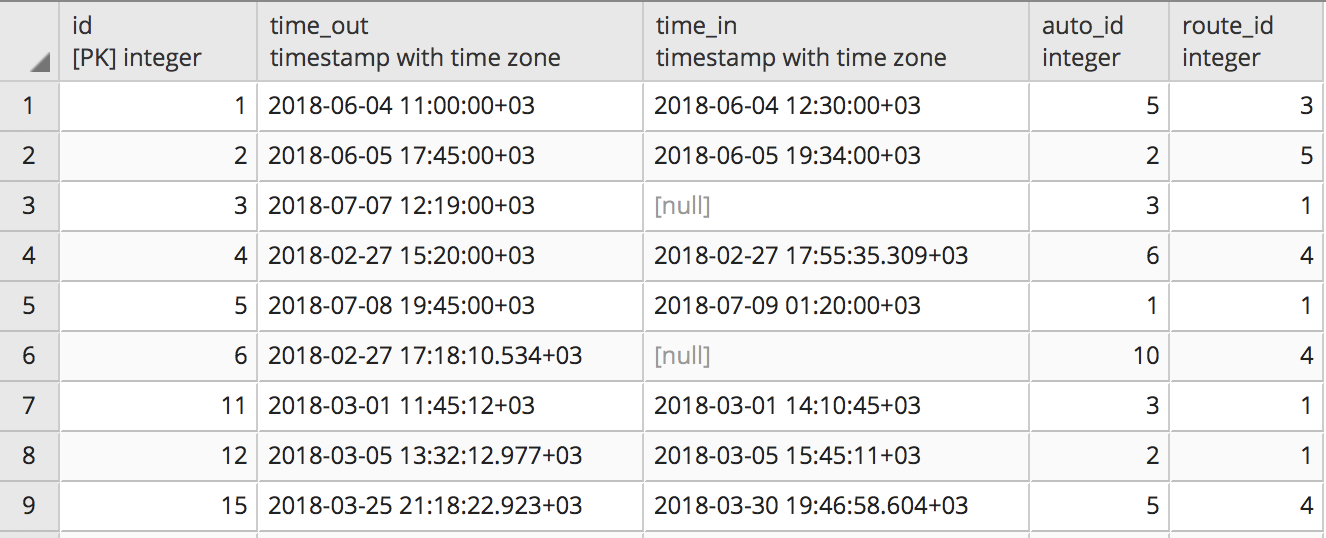
ON journal

for each row

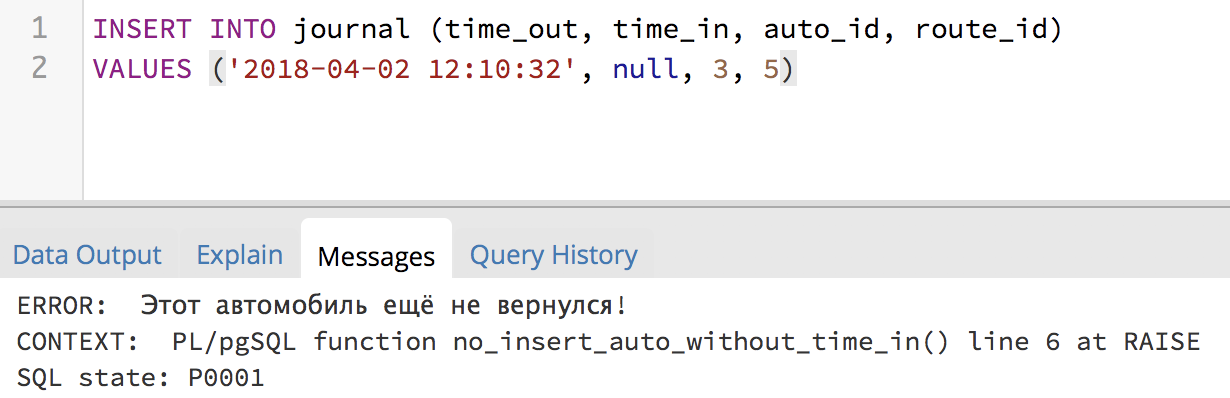
EXECUTE PROCEDURE no\_insert\_auto\_without\_time\_in();

Результаты:

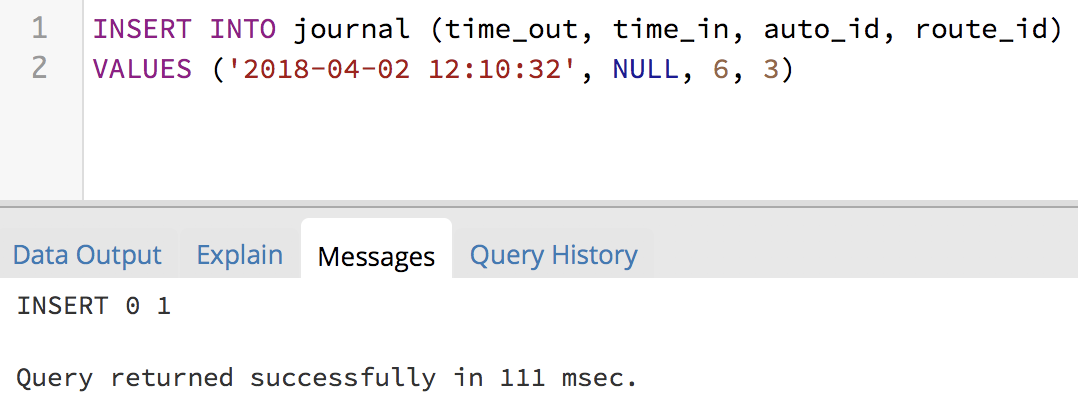
Таблица журнал:



Добавление в журнал автомобиля, который ещё не вернулся:



Добавление в журнал автомобиля, который уже вернулся в парк:



* Триггера на модификацию

1. Создать триггер, не позволяющий изменить номер автомобиля

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION no\_change\_auto\_num() RETURNS trigger AS $first\_update$

BEGIN

IF (OLD.num != NEW.num) then

RAISE 'Нельзя менять номер автомобиля!';

return OLD;

END IF;

END;

$first\_update$ LANGUAGE plpgsql;

CREATE TRIGGER first\_update

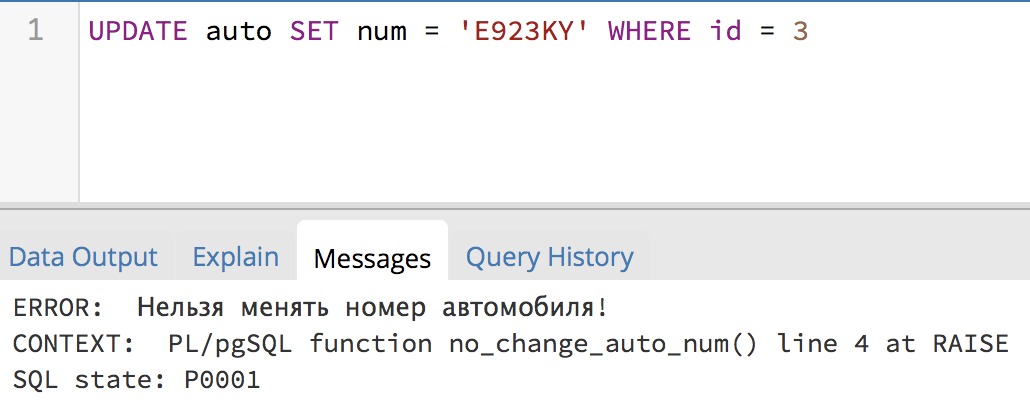
BEFORE UPDATE

ON auto

FOR EACH ROW

EXECUTE PROCEDURE no\_change\_auto\_num();

Изменение номера автомобиля с id = 3:



1. Создать триггер, который не позволит изменить маршрут, в котором есть незаконченные рейсы

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION no\_change\_route\_with\_unfinished\_trip() RETURNS trigger AS $second\_update$

BEGIN

IF (SELECT Count(journal.id) FROM journal, routes WHERE (journal.route\_id = routes.id) AND (journal.time\_in is null) AND (routes.id = OLD.id)) > 0 then

RAISE 'У этого маршрута есть незаконченные рейсы!';

return OLD;

ELSE

return NEW;

END IF;

END;

$second\_update$ LANGUAGE plpgsql;

CREATE TRIGGER second\_update

BEFORE UPDATE

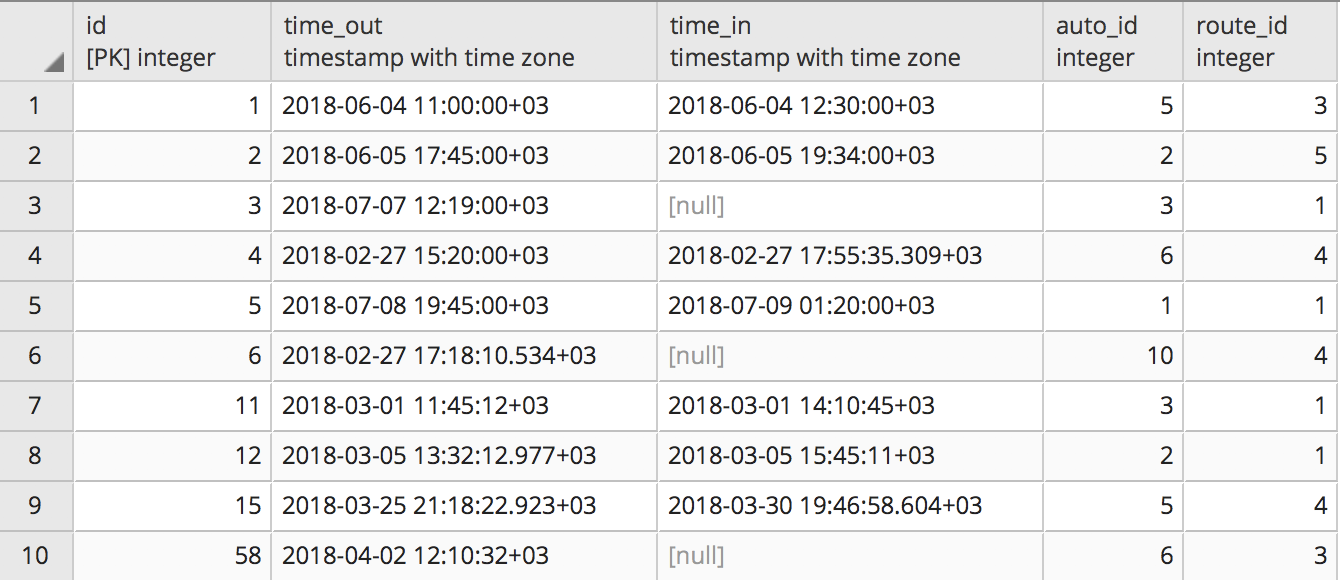
ON routes

FOR EACH ROW

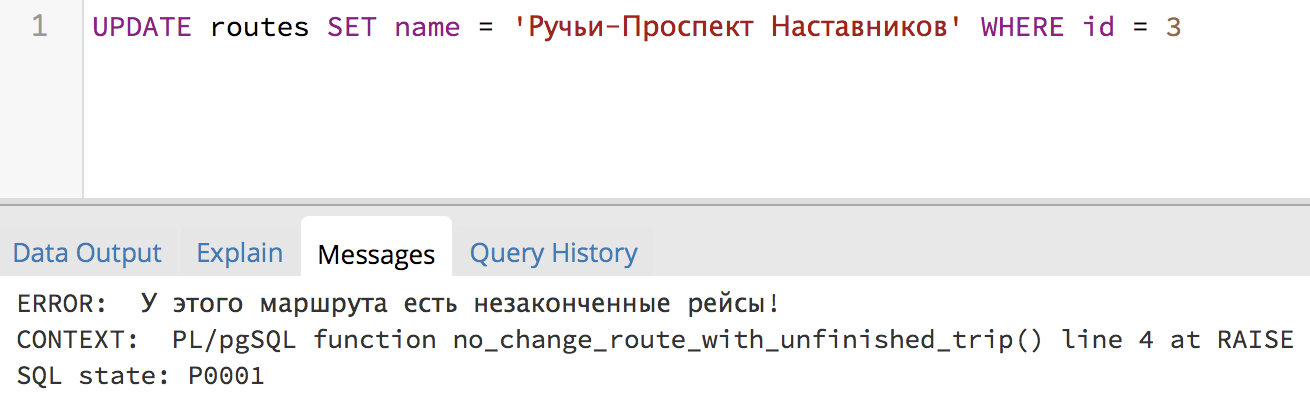
EXECUTE PROCEDURE no\_change\_route\_with\_unfinished\_trip();

Результаты:

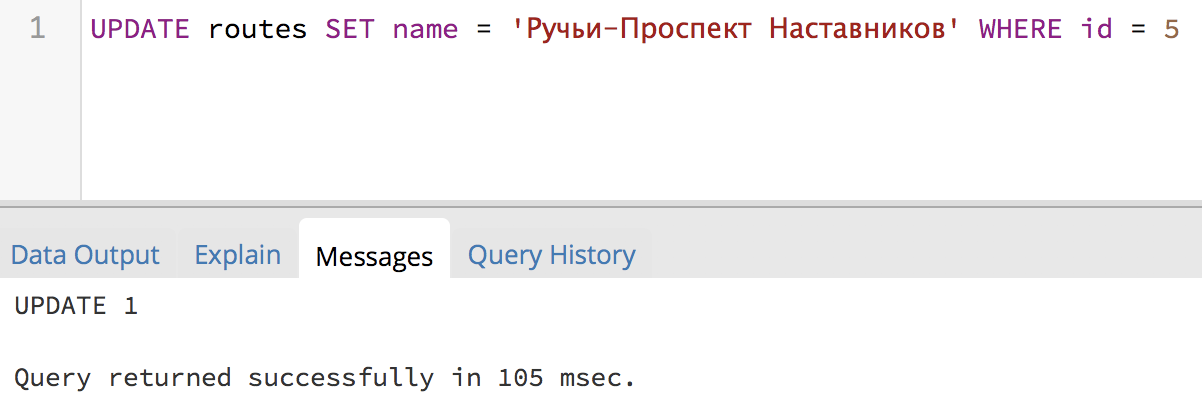
Таблица журнал:



Изменение маршрута с id = 3 (в котором есть незаконченные рейсы):



Изменение маршрута с id = 5:



1. Создать триггер, который при изменении маршрута, в случае наличии ссылок из журнала делает новый маршрут, хранящий неизмененный (прежний) вариант маршрута и подменять все ссылки на него. Таким образом, старые записи журнала будут по-прежнему ссылаться на маршрут, по которому они были созданы.

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION create\_new\_route\_when\_change\_old\_route()

RETURNS trigger AS $third\_update$

DECLARE

change\_id integer;

BEGIN

IF EXISTS(SELECT \* FROM journal WHERE route\_id = OLD.id) THEN

--Заносим в переменную максимальный id маршрута

SELECT MAX(id) INTO change\_id FROM routes;

change\_id = change\_id + 1;

RAISE NOTICE 'Имеются ссылки на этот маршрут!';

--Создаём новый маршрут с максимальным id

INSERT INTO routes(id, name) VALUES(change\_id, NEW.name);

RAISE NOTICE 'Добавлен новый маршрут с id = %', change\_id;

--Переносим все записи таблицы "Журнал" со старого маршрута на новый

UPDATE journal SET route\_id = change\_id WHERE route\_id = NEW.id;

RAISE NOTICE 'Все записи таблица ЖУРНАЛ перенесены со старого маршрута на новый!';

RETURN OLD;

ELSE

RAISE NOTICE 'Маршрут успешно изменён!';

RETURN NEW;

END IF;

END;

$third\_update$ LANGUAGE plpgsql;

CREATE TRIGGER third\_update

BEFORE UPDATE

ON routes

FOR EACH ROW

EXECUTE PROCEDURE create\_new\_route\_when\_change\_old\_route();

Результаты:

Таблица маршрутов:

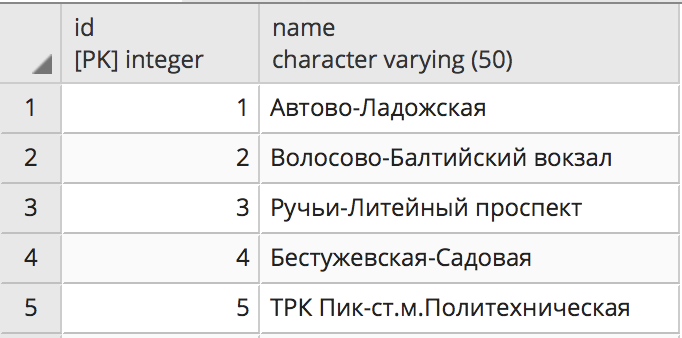
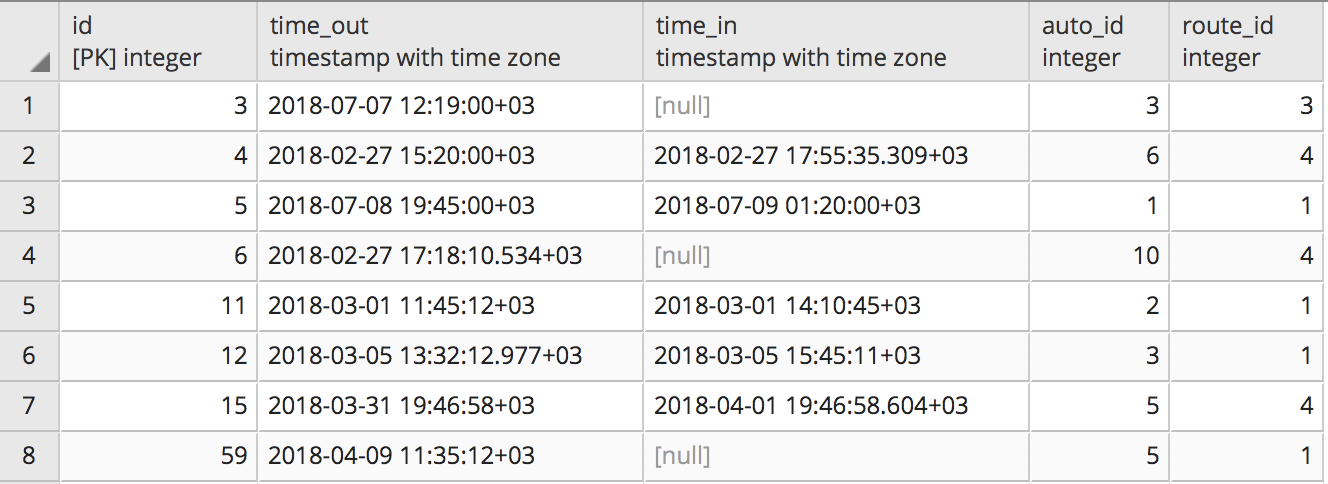
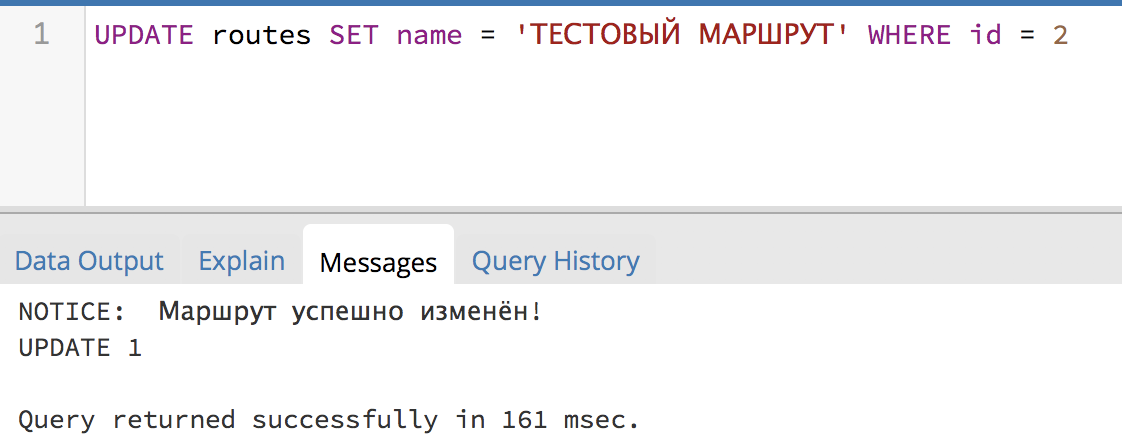
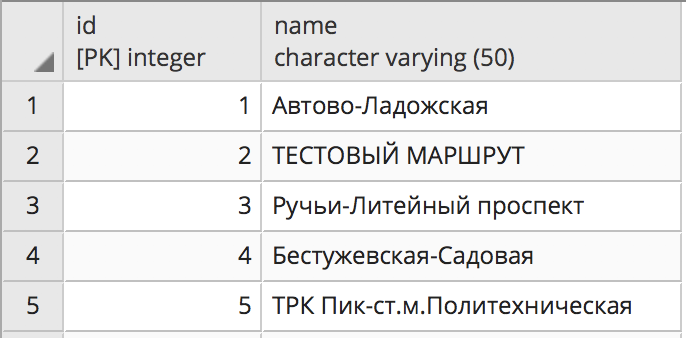


Таблица журнал:

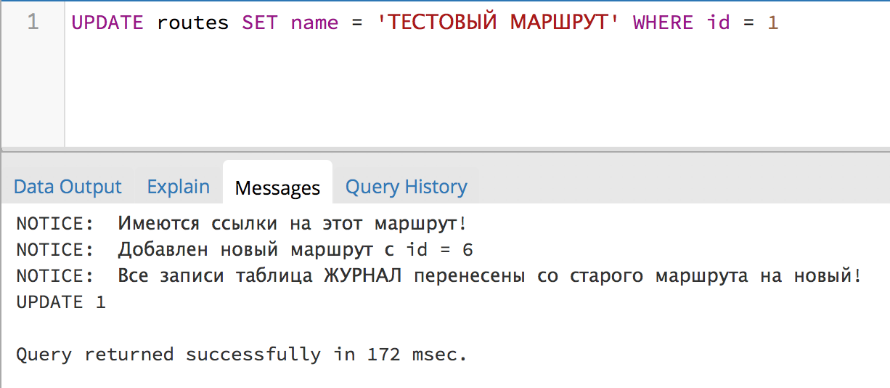


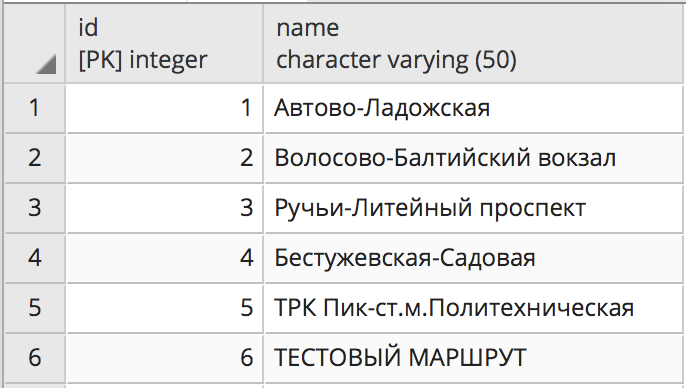
Изменение маршрута с id = 2 (у которого нет записей в журнале):

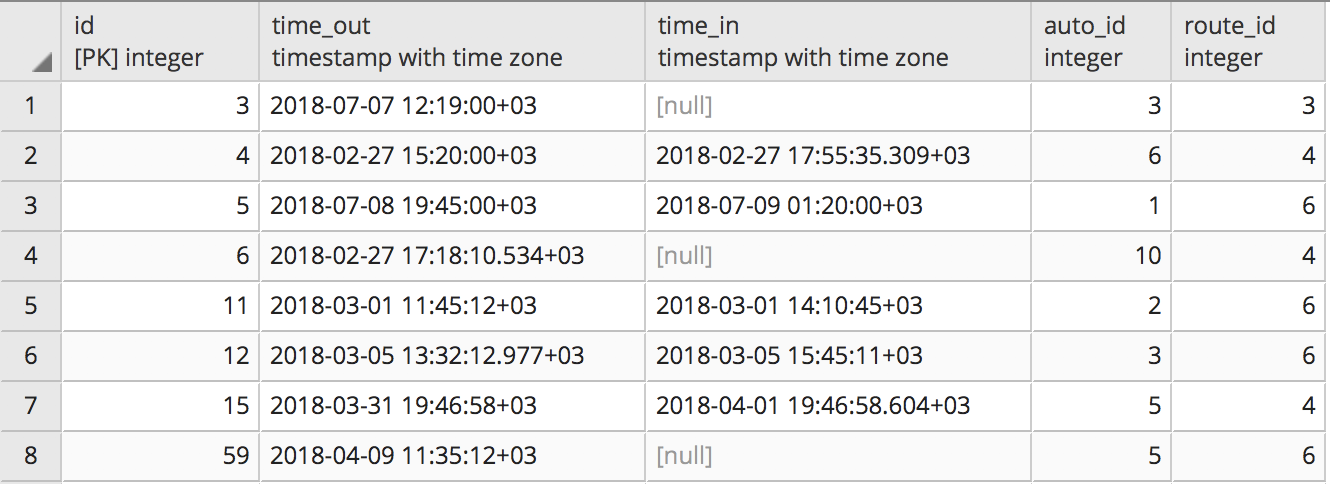




Изменение маршрута с id = 1 (у которого есть записи в журнале):







1. Создать триггер, который не позволит установить время прибытия журнала оператора меньше, чем время отправки

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION no\_time\_in\_less\_than\_time\_out() RETURNS trigger AS $fourth\_update$

BEGIN

IF ((NEW.time\_in < OLD.time\_out) OR (OLD.time\_in < NEW.time\_out)) THEN

RAISE 'Время прибытия не может быть меньше времени отправки!';

return OLD;

ELSE

return NEW;

END IF;

END;

$fourth\_update$ LANGUAGE plpgsql;

CREATE TRIGGER fourth\_update

BEFORE UPDATE

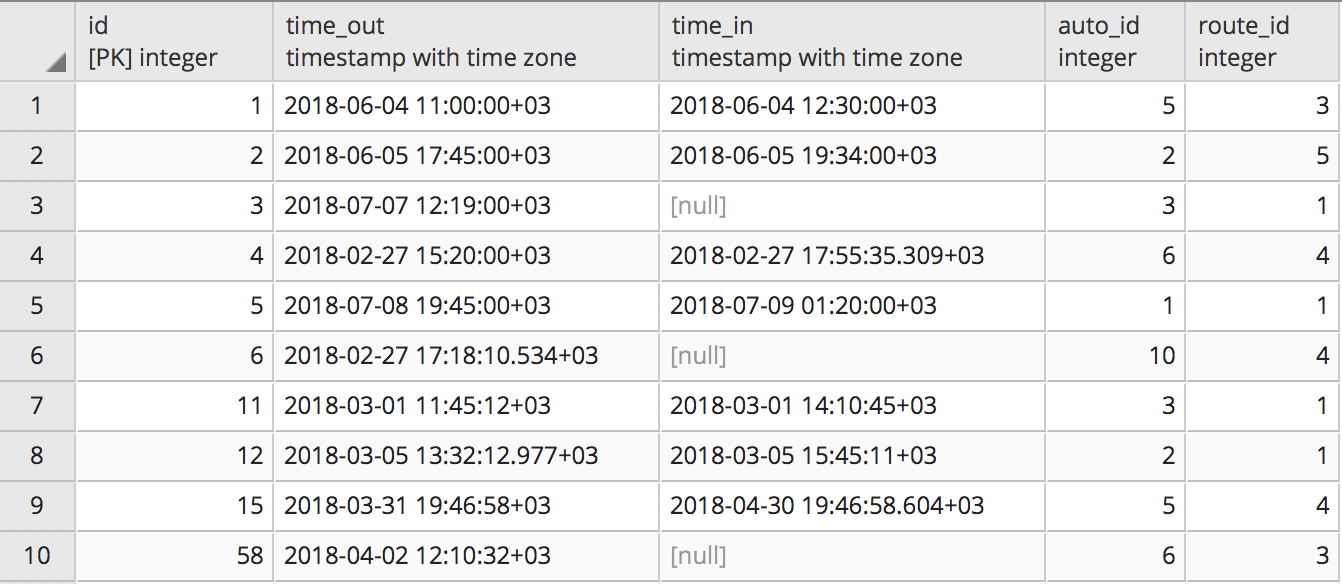
ON journal

FOR EACH ROW

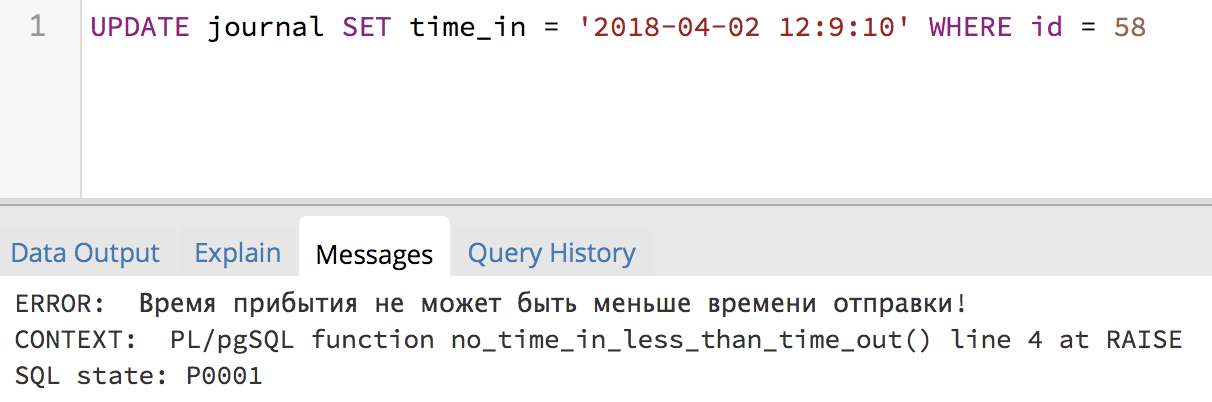
EXECUTE PROCEDURE no\_time\_in\_less\_than\_time\_out();

Результаты:

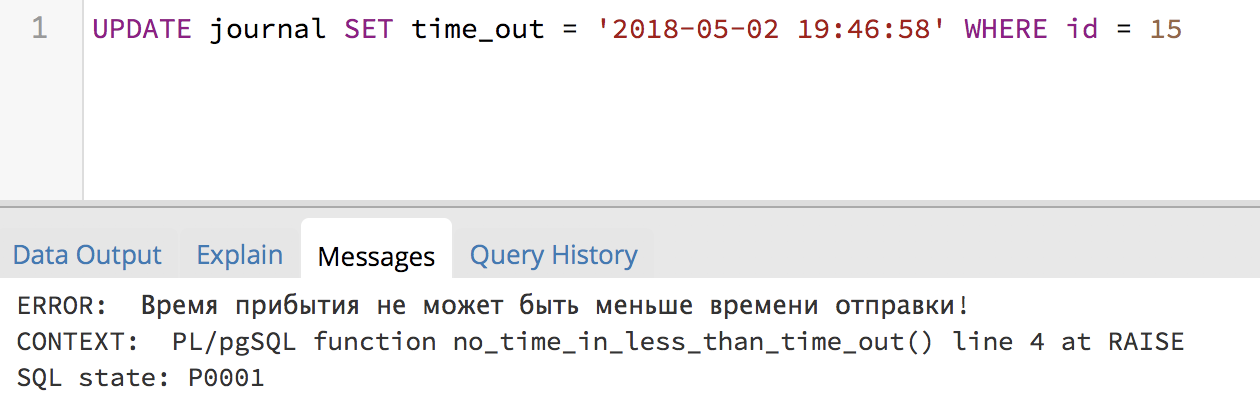
Таблица журнал:



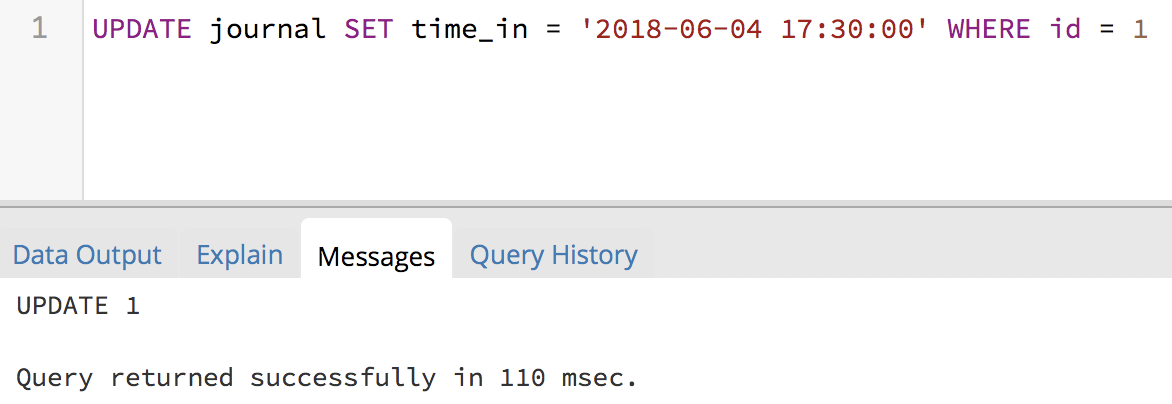
Изменение времени прибытия в записи с id = 58:



Изменение времени отправки в записи с id = 15:



Изменение времени прибытия на не превышающее времени отправки с id = 1:



* Триггера на удаление

1. Создать триггер, который при удалении маршрута в случае наличия на него ссылок откатывает транзакцию

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION delete\_route() RETURNS trigger AS $first\_delete$

BEGIN

IF EXISTS(SELECT \* FROM journal WHERE route\_id = OLD.id) THEN

RAISE exception 'Имеются ссылки на этот маршрут!';

END IF;

RAISE NOTICE 'Маршрут успешно удалён!';

RETURN OLD;

END;

$first\_delete$ LANGUAGE plpgsql;

CREATE TRIGGER first\_delete

BEFORE DELETE

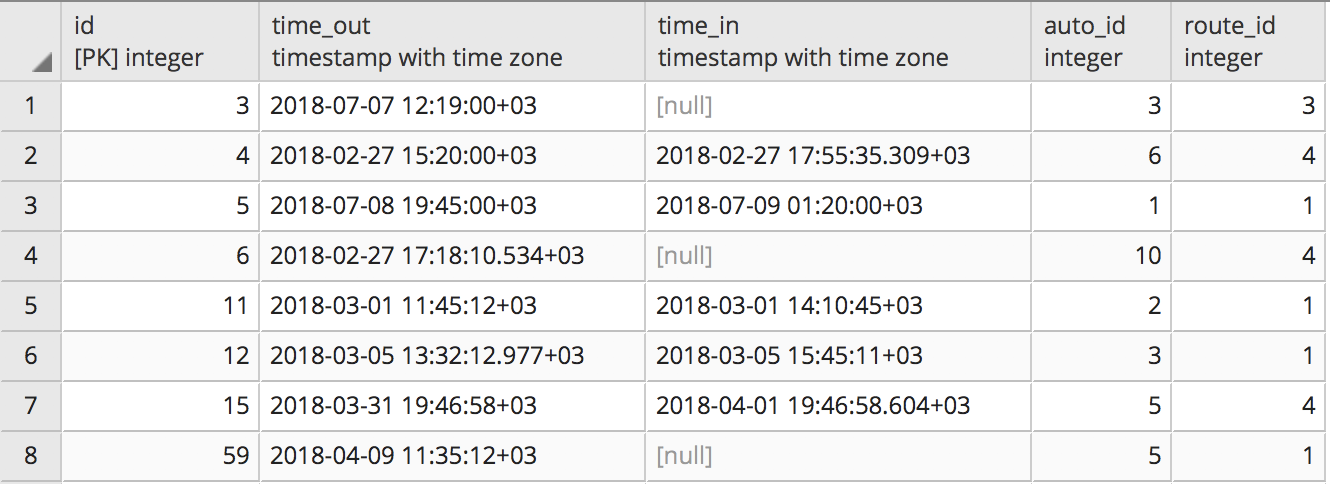
ON routes

FOR EACH ROW

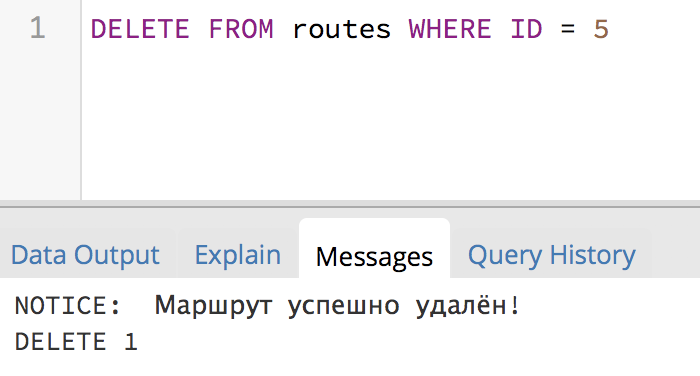
EXECUTE PROCEDURE delete\_route();

Результаты:

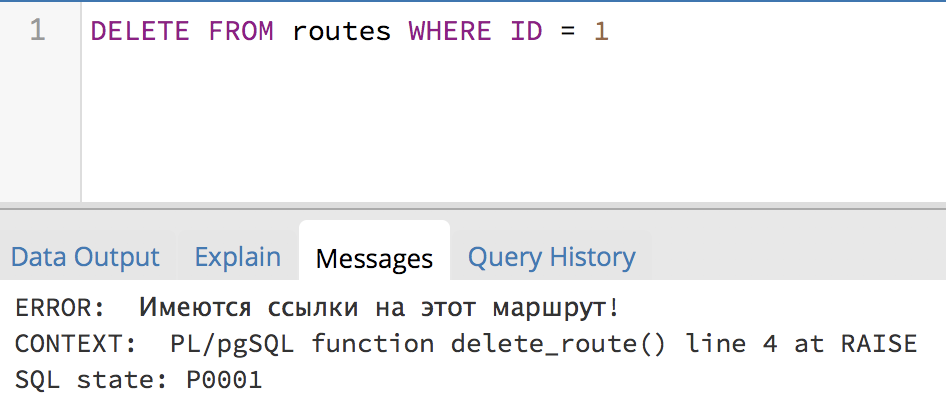
Таблица журнал:



Удаление маршрута с id = 5 (на которого нет ссылок в журнале):



Удаление маршрута с id = 1 (на которого есть ссылки в журнале):



1. Создать триггер, который при удалении автомобиля в случае наличия на него ссылок откатывает транзакцию

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION delete\_auto() RETURNS trigger AS $second\_delete$

BEGIN

IF EXISTS(SELECT \* FROM journal WHERE auto\_id = OLD.id) THEN

RAISE exception 'Имеются ссылки на этот автомобиль!';

END IF;

RAISE NOTICE 'Автомобиль успешно удалён!';

RETURN OLD;

END;

$second\_delete$ LANGUAGE plpgsql;

CREATE TRIGGER second\_delete

BEFORE DELETE

ON auto

FOR EACH ROW

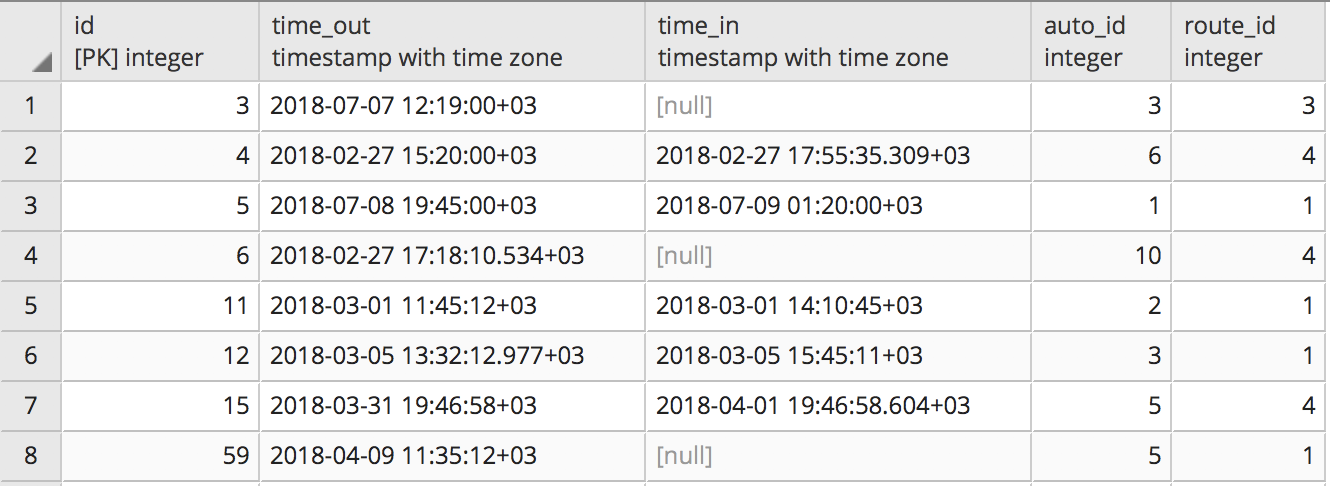
EXECUTE PROCEDURE delete\_auto();

Результаты:

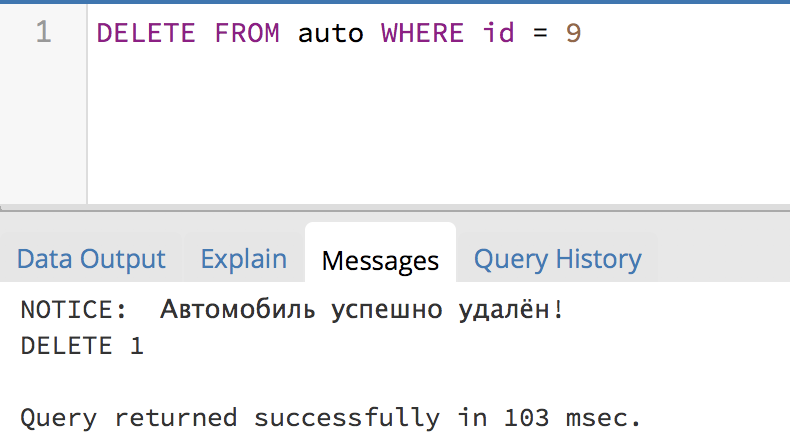
Таблица автомобилей:



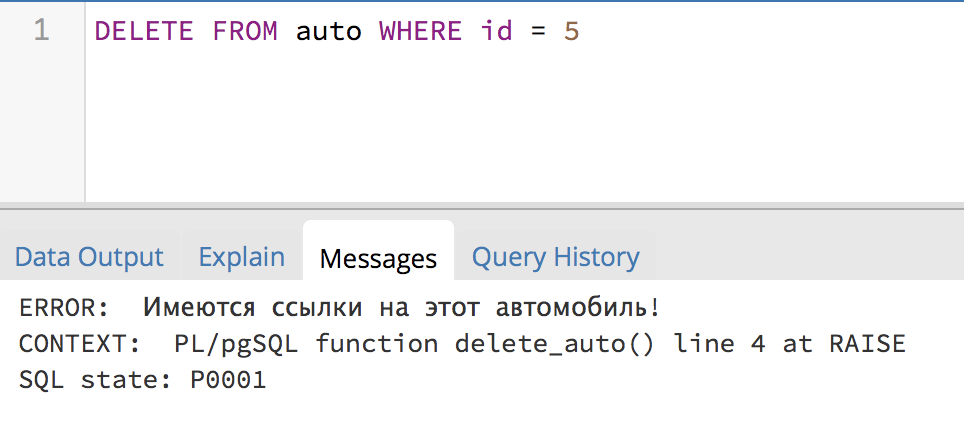
Таблица журнал:



Удаление автомобиля с id = 9 (на которого нет ссылок в журнале):



Удаление автомобиля с id = 5 (на которого есть ссылки в журнале):



1. Создать триггер, который при удалении водителя в случае наличия на него ссылок откатывает транзакцию

-- Создание триггерной функции

CREATE OR REPLACE FUNCTION delete\_auto\_personnel() RETURNS trigger AS $third\_delete$

BEGIN

IF EXISTS(SELECT \* FROM auto WHERE personnel\_id = OLD.id) THEN

RAISE exception 'Имеются ссылки на этого водителя!';

END IF;

RAISE NOTICE 'Водитель успешно удалён!';

RETURN OLD;

END;

$third\_delete$ LANGUAGE plpgsql;

CREATE TRIGGER third\_delete

BEFORE DELETE

ON auto\_personnel

FOR EACH ROW

EXECUTE PROCEDURE delete\_auto\_personnel();

Результаты:

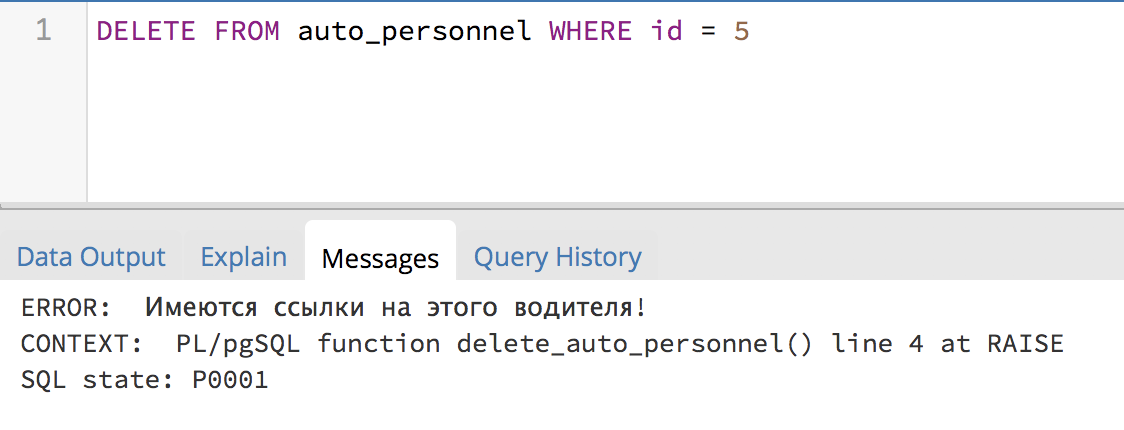
Таблица автомобилей:



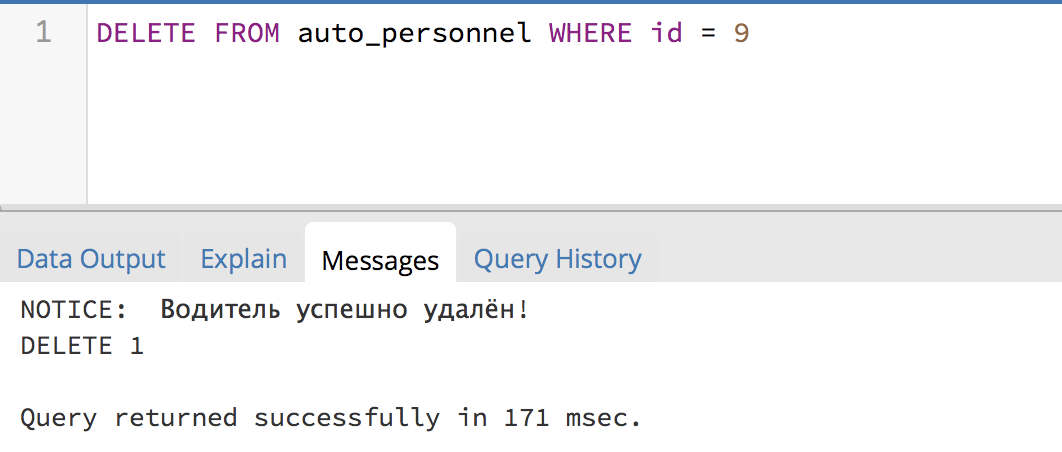
Таблица водителей:



Удаление водителя с id = 5 (на которого есть ссылки в других таблицах):



Удаление водителя с id = 9 (на которого нет ссылок в других таблицах):



Hints: select, where, in, exists, join, commit, rollback, create trigger, drop trigger

### Курсоры

* Хранимая процедура для расчета премий

Необходимо создать хранимую процедуру, распределяющую премии в автопарке. Премиальные денежные средства выдаются трем водителям, проехавшим за минимальное время наибольшее количество маршрутов, причем 50% от премиальной суммы выдается первому водителю, а остальным двум выдается 30% и 20% в соответствии с их успехами.

Хранимая процедура должна иметь три входных параметра: дату начала, дату окончания и сумму премиальных, подлежащих распределению за требуемый период времени. Результатом работы хранимой процедуры должны быть три водителя с суммами их премиальных.

Предлагаемый алгоритм следующий: создаем 3 переменные для хранения идентификаторов водителей и 3 переменные для хранения их результатов. Создаем курсор, который пробегает по таблице персонала и, если автомобили данного водителя поставили рекорд в большем числе маршрутов, нежели уже выбранные водители, хранящиеся в переменных, то данный водитель вытесняет ранее сохраненного водителя с худшими результатами. Результатом работы хранимой процедуры будет выборка значений переменных с суммами, рассчитанными по описанному выше алгоритму.

CREATE OR REPLACE FUNCTION func3(IN var3 integer, IN var1 text, IN var2 text)

RETURNS text

AS $$

txt = ""

dr1 = ""

dr2 = ""

dr3 = ""

a1 = 0

a2 = 0

a3 = 0

for row1 in plpy.cursor("select id from auto\_personnel ;"):

txt = str(row1['id'])

for row2 in plpy.cursor("select auto\_personnel.first\_name, auto\_personnel.last\_name,(select count(\*) from journal join auto on journal.auto\_id = auto.id join auto\_personnel on auto.personnel\_id = auto\_personnel.id where auto\_personnel.id = " + txt + " ) as count from auto\_personnel join auto on auto\_personnel.id = auto.personnel\_id join journal on auto.id = journal.auto\_id where journal.id = " + txt + ";"):

if a1 < row2['count']:

dr3 = dr2

dr2 = dr1

dr1 = row2['first\_name'] + " " + row2['last\_name']

a3 = a2

a2 = a1

a1 = row2['count']

elif a2 < row2['count']:

dr3 = dr2

dr2 = row2['first\_name'] + " " + row2['last\_name']

a3 = a2

a2 = row2['count']

elif a3 < row2['count']:

dr3 = row2['first\_name'] + " " + row2['last\_name']

a3 = row2['count']

a3 = var3 \* 0.2

a2 = var3 \* 0.3

a1 = var3 \* 0.5

txt = dr1 + " " + str(a1) + " | " + dr2 + " " + str(a2) + " | " + dr3 + " " + str(a3)

return txt

$$

LANGUAGE plpython3u;

SELECT \* FROM func3(13400, '2017-12-12', '2020-12-12');

Результат:

