**Цель работы**: Научиться создавать хранимые процедуры, функции и триггеры в процедурных расширениях SQL.

**Задание :**

Для баз данных, созданных в лабораторной работе №3 создать:

* Хранимую процедуру с параметрами и хранимую функцию с параметрами, реализующие бизнес – правила предметной области курсовой работы (MySQL).
* Функции sql с параметрами, бизнес – правила предметной области курсовой работы (PostgreSQL).
* Триггеры для событий INSERT, UPDATE и DELETE, реализующие нестандартные правила обеспечения ссылочной целостности базы данных предметной области курсовой работы (MySQL и PostgreSQL).

Создадим процедуру ispolniteli, которая выводит информацию об исполнителях (псевдоним, возраст и амплуа) заданной группы.

BEGIN

select `musikalnaya\_gruppa`.`Nazvanie\_gruppi`, `ispolnitel`.`Psevdonim/Familia`, `ispolnitel`.`Vozrast`, `ispolnitel`.`Amplua`

from (`ispolnitel` join `musical\_group\_ispolnitel` on `musical\_group\_ispolnitel`.`Identifikazionnij\_nomer\_ispolnitelya` = `ispolnitel`.`Identifikazionnij\_nomer\_ispolnitelya`) join `musikalnaya\_gruppa` on `musical\_group\_ispolnitel`.`Nazvanie\_gruppi` = `musikalnaya\_gruppa`.`Nazvanie\_gruppi`

where musikalnaya\_gruppa.Nazvanie\_gruppi = Nazvanie\_g;

END

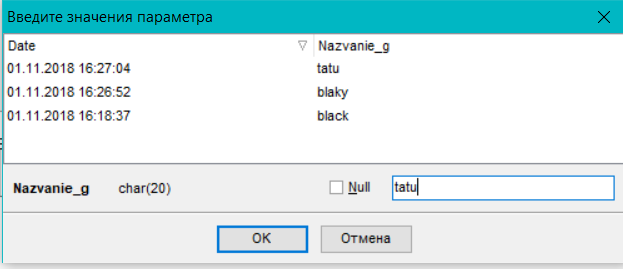


Рисунок 1 — диалоговое окно процедуры

Рисунок 2 — результат выполнения процедуры

Создадим функцию, которая выводит цену билета на последний концерт указанной группы.

BEGIN

DECLARE Var BIGINT DEFAULT 0;

select `zena`.`Zena\_bileta` INTO Var

from `zena` join `vistuplenie`on zena.`Mesto\_vistupleniya`= `vistuplenie`.`Mesto\_vistupleniya`

where Data\_vistuplenia =

(select MAX(`vistuplenie`.`Data\_vistuplenia`)

from (`mesto\_v\_poslednem\_hit\_parade` join zena on `mesto\_v\_poslednem\_hit\_parade`.`Nomer\_v\_hit\_parade` = `zena`.`Nomer\_v\_hit\_parade`) join `vistuplenie` on `vistuplenie`.`Mesto\_vistupleniya` = zena.`Mesto\_vistupleniya`

where `vistuplenie`.`Nazvanie\_gruppi` = Nazvanie\_g

);

RETURN Var;

END



Рисунок 3 — таблица «выступление»

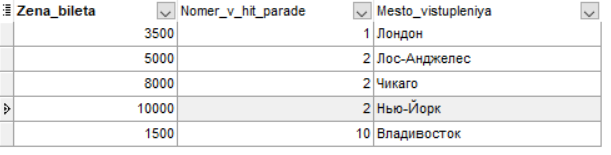


Рисунок 4 — таблица «цена»

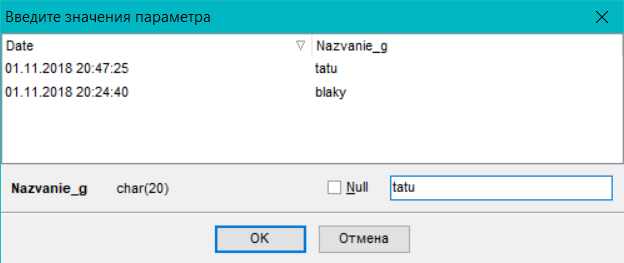


Рисунок 5 - диалоговое окно функции song



Рисунок 6 — результат выполнения функции

Для создания триггеров воспользуемся SQL-скриптом в EMS SQL Manager for MySQL для базы данных «Музыкальные группы».

Создадим триггеры для событий UPDATE и DELETE в таблице "Музыкальная\_группа\_Исполнитель". Исходя из бизнес-правила курсовой работы, при удалении исполнителя или обновлении внешнего ключа, мы не можем оставить группу без единого исполнителя. В любой группе должен быть, как минимум, один исполнитель.

Триггер на событие UPDATE в таблице «Музыкальная\_группа\_исполнитель» :

CREATE DEFINER = 'root'@'localhost' TRIGGER `mgi\_before\_update\_tr1` BEFORE UPDATE ON `musical\_group\_ispolnitel`

FOR EACH ROW

BEGIN

DECLARE ERROR CONDITION FOR SQLSTATE '45000';

SET @p=(SELECT COUNT(OLD.Nazvanie\_gruppi)

FROM `musical\_group\_ispolnitel`

WHERE Nazvanie\_gruppi=OLD.Nazvanie\_gruppi);

IF (@p=1)THEN

signal ERROR set message\_text ='V gruppe odin ispolnitel! Obnovlenie zapreshcheno!!!';

END IF;

END;

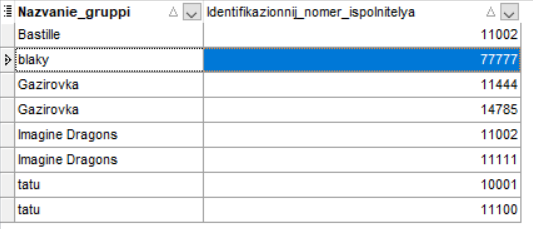


Рисунок 7 - в группе blaky всего один исполнитель

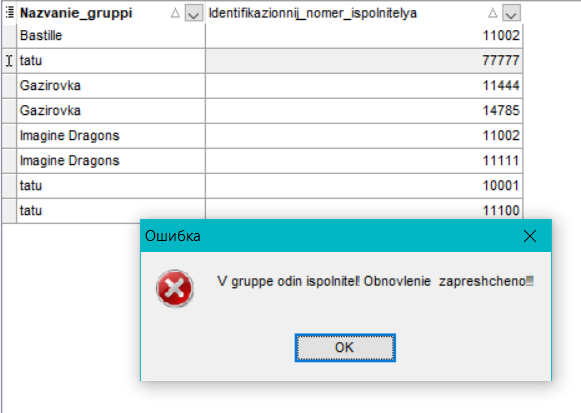


Рисунок 8 — сообщение об ошибке обновления внешнего ключа

Триггер на событие DELETE в таблице «Музыкальная\_группа\_исполнитель» :

CREATE DEFINER = 'root'@'localhost' TRIGGER `mgi\_before\_del\_tr1` BEFORE DELETE ON `musical\_group\_ispolnitel`

FOR EACH ROW

BEGIN

DECLARE ERROR CONDITION FOR SQLSTATE '45000';

SET @p=(SELECT COUNT(OLD.Nazvanie\_gruppi)

FROM `musical\_group\_ispolnitel`

WHERE Nazvanie\_gruppi=OLD.Nazvanie\_gruppi);

IF (@p=1)THEN

signal ERROR set message\_text ='V gruppe odin ispolnitel! Udalyat ego zapreshcheno!!!';

END IF;

END;

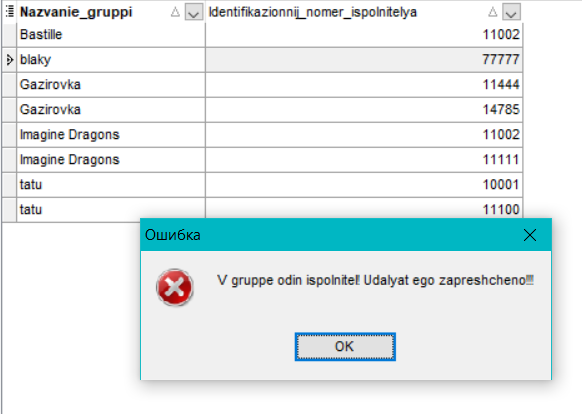


Рисунок 9 — сообщение об ошибке удаления единственного исполнителя из группы blaky

Согласно бизнес-правилу предметной области «Музыкальные группы», если мы создаем новую музыкальную группу, то для нее обязательно должна быть создана песня по умолчанию в таблице «Песня», так как музыкальная группа должна содержать, как минимум, одну песню.

Триггер на событие INSERT в таблице «Музыкальная\_группа» :

CREATE DEFINER = 'root'@'localhost' TRIGGER `musikgruppa\_after\_ins\_tr1` AFTER INSERT ON `musikalnaya\_gruppa`

FOR EACH ROW

BEGIN

insert into `pesnya` (Nazvanie\_pesni, Avtor\_teksta, Kompositor, Data\_sozdania, Yazik\_pesni, Nazvanie\_gruppi) values ('Новая песня', 'Новый автор текста', 'Новый композитор', null, null, NEW.Nazvanie\_gruppi);

END;

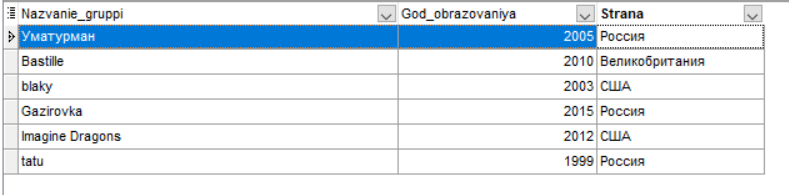


Рисунок 10 — создание новой группы

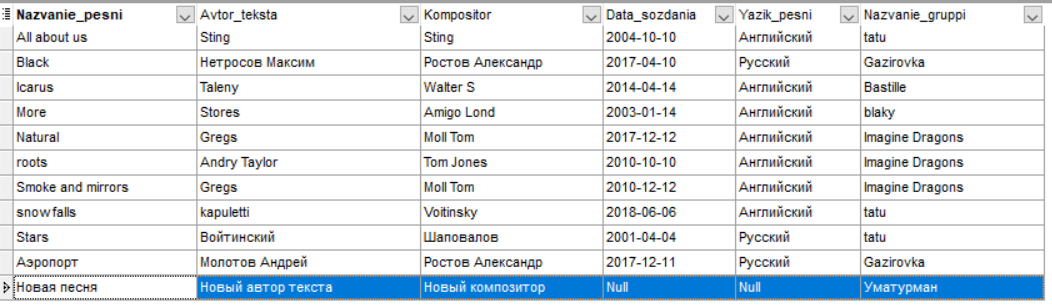


Рисунок 11 — в таблице «Песня» создана строка со значениями по умолчанию для созданной группы

Согласно бизнес-правилу предметной области, удалять и обновлять внешний ключ песни, если она является единственной у музыкальной группы, запрещено.

Триггер на событие UPDATE в таблице «Песня» :

CREATE DEFINER = 'root'@'localhost' TRIGGER `pesnya\_before\_update\_tr1` BEFORE UPDATE ON `pesnya`

FOR EACH ROW

BEGIN

DECLARE ERROR CONDITION FOR SQLSTATE '45000';

SET @p=(SELECT COUNT(OLD.Nazvanie\_gruppi)

FROM `pesnya`

WHERE Nazvanie\_gruppi=OLD.Nazvanie\_gruppi);

IF (@p=1)THEN

signal ERROR set message\_text ='Musikalnaya gruppa sodergit odnu pesnu. Obnovlyat gruppu zapreshcheno!';

END IF;

END;

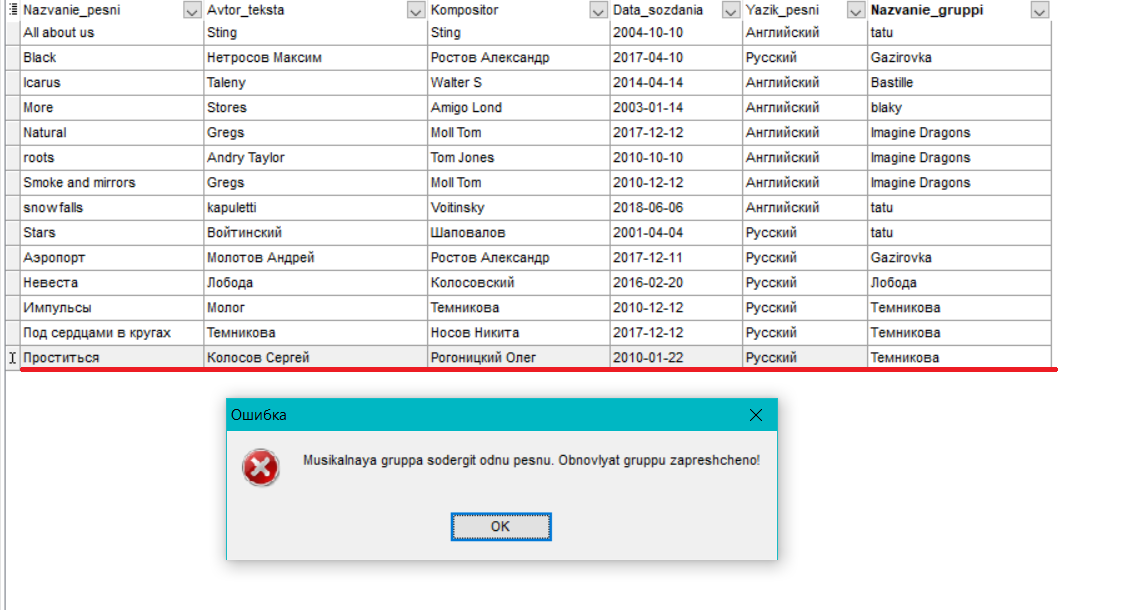


Рисунок 12 — сообщение об ошибке при обновлении внешнего ключа у строки в таблице «Песня»

Триггер на событие DELETE в таблице «Песня» :

CREATE DEFINER = 'root'@'localhost' TRIGGER `pesnya\_before\_del\_tr1` BEFORE DELETE ON `pesnya`

FOR EACH ROW

BEGIN

DECLARE ERROR CONDITION FOR SQLSTATE '45000';

SET @p=(SELECT COUNT(OLD.Nazvanie\_gruppi)

FROM `pesnya`

WHERE Nazvanie\_gruppi=OLD.Nazvanie\_gruppi);

IF (@p=1)THEN

signal ERROR set message\_text ='V gruppe odna pesnya! Udalyat ee zapreshcheno!!!';

END IF;

END;

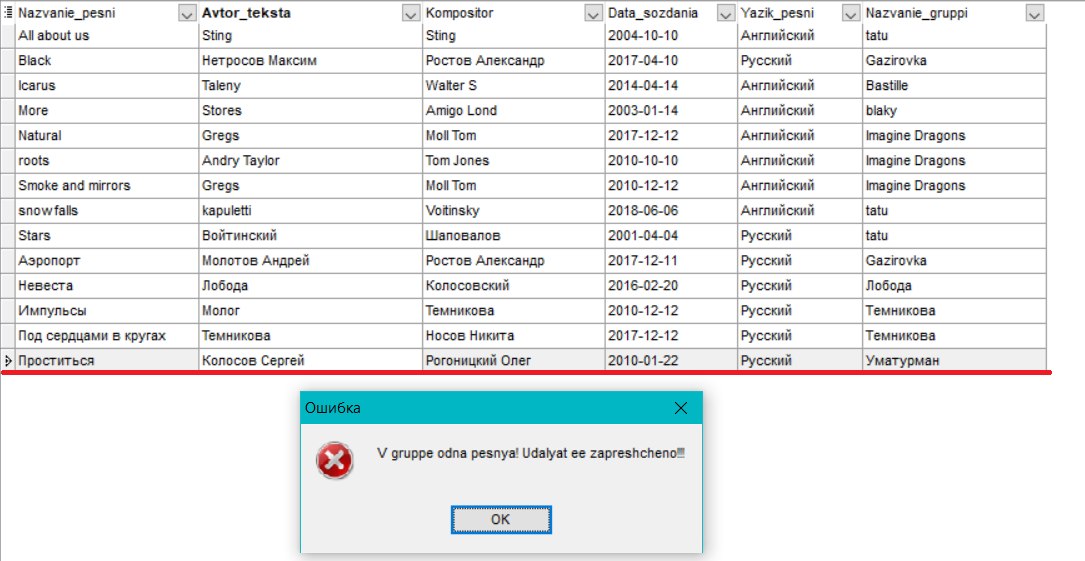


Рисунок 13 — сообщение об ошибке при удалении строки из таблицы «Песня»

Бизнес-правило предметной области «Музыкальная группа» : нельзя удалять выступление, если оно является единственным в гастролях, так как гастроли должны содержать, как минимум, одно выступление.

Триггер на события DELETE в таблице «Выступление» :

CREATE DEFINER = 'root'@'localhost' TRIGGER `vistuplenie\_before\_del\_tr1` BEFORE DELETE ON `vistuplenie`

FOR EACH ROW

BEGIN

DECLARE ERROR CONDITION FOR SQLSTATE '45000';

SET @p=(SELECT COUNT(OLD.Nazvanie)

FROM `vistuplenie`

WHERE Nazvanie=OLD.Nazvanie);

IF (@p=1)THEN

signal ERROR set message\_text ='V gastrolyah odno vistuplenie! Udalyat nelzya!';

END IF;

END;

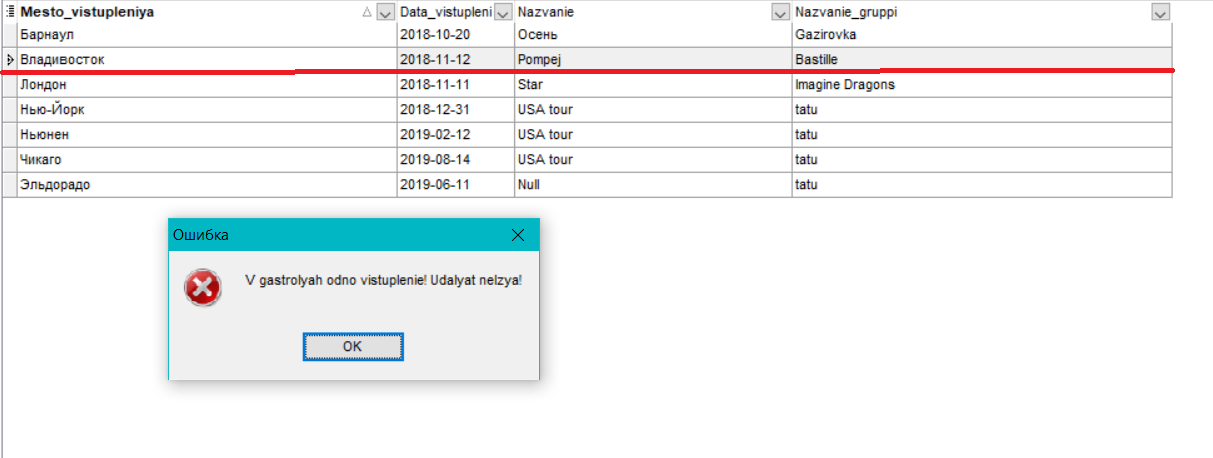


Рисунок 14 - сообщение об ошибке при удалении выступления

Создадим триггер, который при добавлении номера места в хит-параде проверяет, не превышает ли он 100 (хит-парад содержит всего 100 мест).

CREATE DEFINER = 'root'@'localhost' TRIGGER `mesto\_v\_hit\_parade\_after\_ins\_tr1` AFTER INSERT ON `mesto\_v\_poslednem\_hit\_parade`

FOR EACH ROW

BEGIN

DECLARE ERROR CONDITION FOR SQLSTATE '45000';

IF (NEW.Nomer\_v\_hit\_parade>100)

then signal ERROR set message\_text = 'Hit parad sodergit vsego 100 mest!';

END IF;

END;

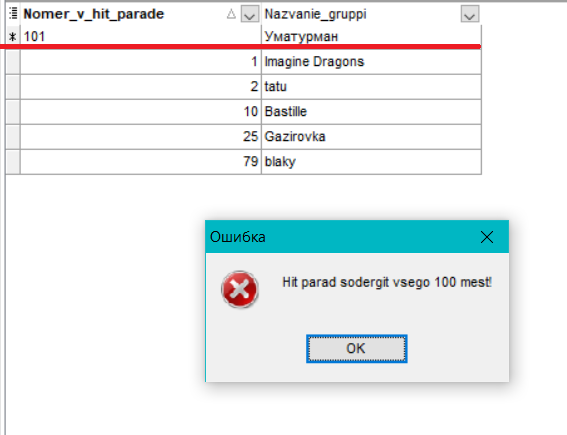


Рисунок 15 — сообщение об ошибке при добавлении номера в хит-параде, превышающего 100

Бизнес-правило предметной области : цена билета на концерт группы зависит от номера в хит-параде и места выступления. Если группа находится в ТОП-5 хит-парада, то цена билета должна быть не менее 10000. Если место выступления группы является город Чикаго или Лондон, или Москва — не менее 15000, так как в данных городах билеты на концерт, в среднем, являются самыми дорогими.

Триггер на событие INSERT в таблице «Цена» :

CREATE DEFINER = 'root'@'localhost' TRIGGER `nomer\_and\_zena` AFTER INSERT ON `zena`

FOR EACH ROW

BEGIN

DECLARE ERROR CONDITION FOR SQLSTATE '45000';

IF (NEW.Nomer\_v\_hit\_parade<=5 and NEW.Zena\_bileta<10000)

then signal ERROR set message\_text ='Dannaya gruppa vhodit v TOP-5 hit-parada . Zena bileta dolzhna bit ne menee 10000!';

END IF;

IF(NEW.Mesto\_vistupleniya='Лондон' or NEW.Mesto\_vistupleniya='Чикаго' or NEW.Mesto\_vistupleniya='Москва' and NEW.Zena\_bileta<15000)

then signal ERROR set message\_text ='Dannaya gruppa vistupaet v samom dorogom gorode . Zena bileta dolzhna bit ne menee 15000!';

END IF;

END;

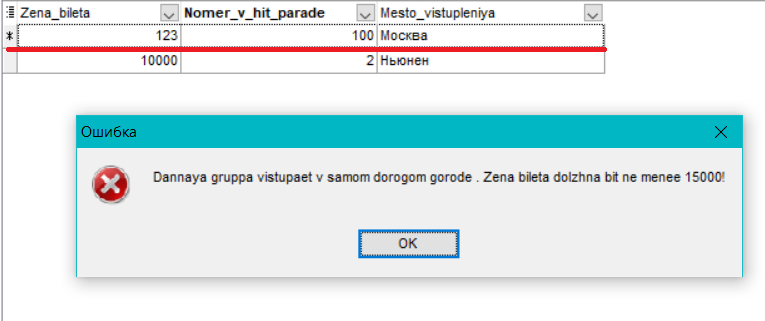


Рисунок 16 — сообщение об ошибке при добавлении цены для выступления, которое проходит в самом дорогом городе

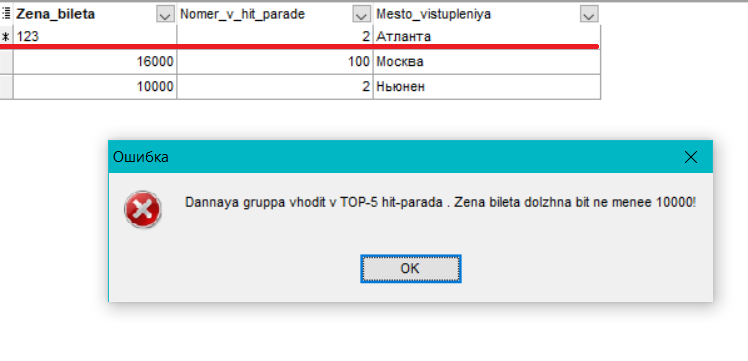


Рисунок 17 — сообщение об ошибке при добавлении цены для выступления группы, которая находится в ТОП-5 хит-парада

Для базы данных, созданной в лабораторной работе №3 создадим функцию sql с параметрами, бизнес – правила предметной области курсовой работы (PostgreSQL).

Создадим функцию, которая возвращает одно значение - наибольшую цену билета на концерт.

DECLARE

var bigint;

BEGIN

select "Zena"."Zena\_bileta" INTO var

from ("Vistuplenie" join "Zena" on "Vistuplenie"."Mesto\_vistupleniya" = "Zena"."Mesto\_vistupleniya") join "Musikalnaya\_gruppa" on "Vistuplenie"."Nazvanie\_gruppi" = "Musikalnaya\_gruppa"."Nazvanie\_gruppi"

where "Vistuplenie"."Data\_vistuplenia" =

(select MAX("Vistuplenie"."Data\_vistuplenia")

from ("Vistuplenie" join "Zena" on "Vistuplenie"."Mesto\_vistupleniya" = "Zena"."Mesto\_vistupleniya") join "Musikalnaya\_gruppa" on "Vistuplenie"."Nazvanie\_gruppi" = "Musikalnaya\_gruppa"."Nazvanie\_gruppi"

);

RETURN var;

END;

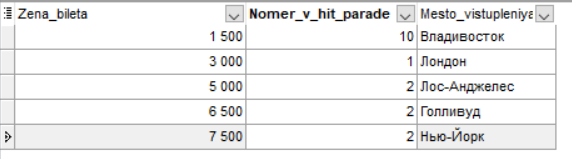


Рисунок 7 — таблица «цена»

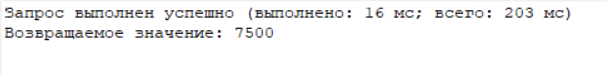


Рисунок 8 — результат выполнения функции

Создадим функцию, которая выводит таблицу - список исполнителей и их принадлежность к группе, возраст которых меньше заданного.

select "Musikalnaya\_gruppa"."Nazvanie\_gruppi", "Ispolnitel"."Psevdonim\_Familia"

from ("Musikalnaya\_gruppa" join "Musikalnaya\_gruppa\_ispolnitel" on "Musikalnaya\_gruppa"."Nazvanie\_gruppi" = "Musikalnaya\_gruppa\_ispolnitel"."Nazvanie\_gruppi")

join "Ispolnitel" on "Ispolnitel"."Identifikazionnij\_nomer\_ispolnitelya" = "Musikalnaya\_gruppa\_ispolnitel"."Identifikazionnij\_nomer\_ispolnitelya"

where "Ispolnitel"."Vozrast"<=$1;



Рисунок 9 — параметры функции

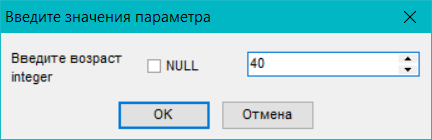


Рисунок 10 — диалоговое окно ввода аргумента



Рисунок 11 — результат выполнения функции

Создадим функцию sql с параметрами, возвращающую столбец текстовых значений. Функция f1 выводит список исполнителей, чей возраст не превышает заданного.

select "Ispolnitel"."Psevdonim\_Familia"

from ("Musikalnaya\_gruppa" join "Musikalnaya\_gruppa\_ispolnitel" on "Musikalnaya\_gruppa"."Nazvanie\_gruppi" = "Musikalnaya\_gruppa\_ispolnitel"."Nazvanie\_gruppi")

join "Ispolnitel" on "Ispolnitel"."Identifikazionnij\_nomer\_ispolnitelya" = "Musikalnaya\_gruppa\_ispolnitel"."Identifikazionnij\_nomer\_ispolnitelya"

where "Ispolnitel"."Vozrast"<=$1;

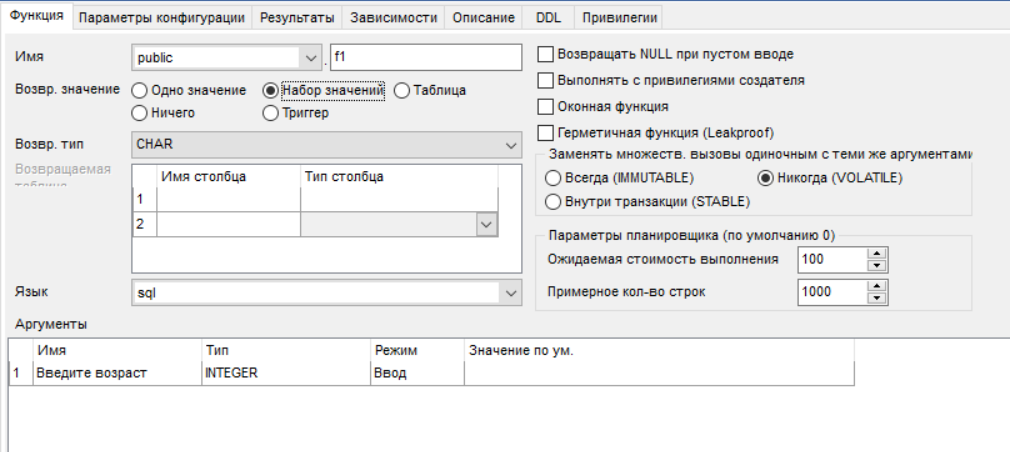


Рисунок 12 - параметры функции

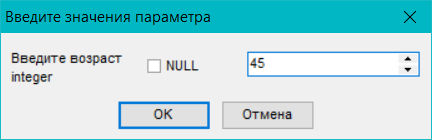


Рисунок 13 — диалоговое окно ввода аргумента

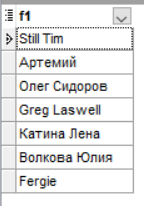


Рисунок 14 — результат выполнения функции

Для создания триггеров воспользуемся SQL-скриптом в EMS SQL Manager for PostgreSQL для базы данных «Музыкальные группы».

Создадим триггеры для событий UPDATE и DELETE в таблице "Музыкальная\_группа\_Исполнитель". Исходя из бизнес-правила курсовой работы, при удалении исполнителя или обновлении внешнего ключа, мы не можем оставить группу без единого исполнителя. В любой группе должен быть, как минимум, один исполнитель.

Триггер на событие UPDATE в таблице «Музыкальная\_группа\_исполнитель» :

CREATE FUNCTION mgi\_before\_update() RETURNS trigger LANGUAGE plpgsql AS $$

DECLARE p integer;

BEGIN

SELECT COUNT(OLD."Nazvanie\_gruppi")

FROM "Musikalnaya\_gruppa\_ispolnitel" into p

WHERE "Nazvanie\_gruppi" =OLD."Nazvanie\_gruppi";

IF (p=1)THEN raise exception 'Данная группа имеет одного исполнителя! ОБНОВЛЕНИЕ запрещено!';

END IF;

return OLD;

END;

$$;

CREATE TRIGGER mgi\_update

BEFORE UPDATE ON "Musikalnaya\_gruppa\_ispolnitel" FOR EACH ROW

EXECUTE PROCEDURE mgi\_before\_update()

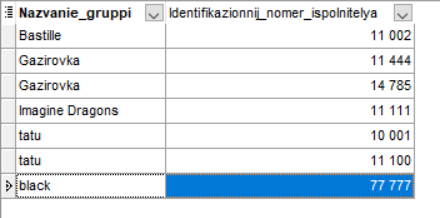


Рисунок 15 — в группе black один исполнитель

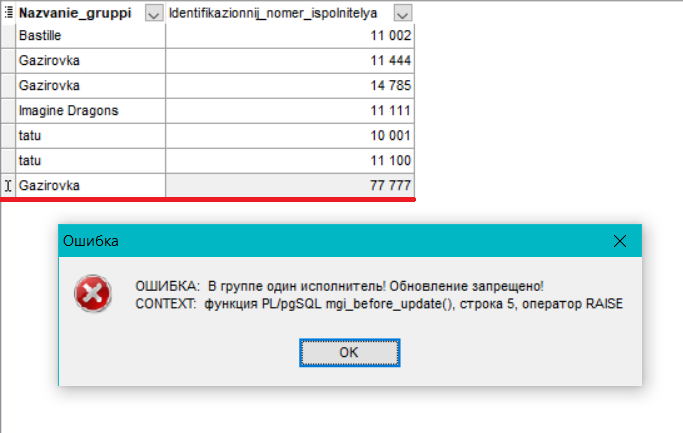


Рисунок 16 — сообщение об ошибке при обновлении внешнего ключа

Триггер на событие DELETE в таблице «Музыкальная\_группа\_исполнитель» :

CREATE FUNCTION mgi\_before\_delete() RETURNS trigger LANGUAGE plpgsql AS $$

DECLARE p integer;

BEGIN

SELECT COUNT(OLD."Nazvanie\_gruppi")

FROM "Musikalnaya\_gruppa\_ispolnitel" into p

WHERE "Nazvanie\_gruppi" =OLD."Nazvanie\_gruppi";

IF (p=1)THEN raise exception 'Данная группа имеет одного исполнителя! Удаление запрещено!';

END IF;

return OLD;

END;

$$;

CREATE TRIGGER mgi\_delete

BEFORE DELETE ON "Musikalnaya\_gruppa\_ispolnitel" FOR EACH ROW

EXECUTE PROCEDURE mgi\_before\_delete()

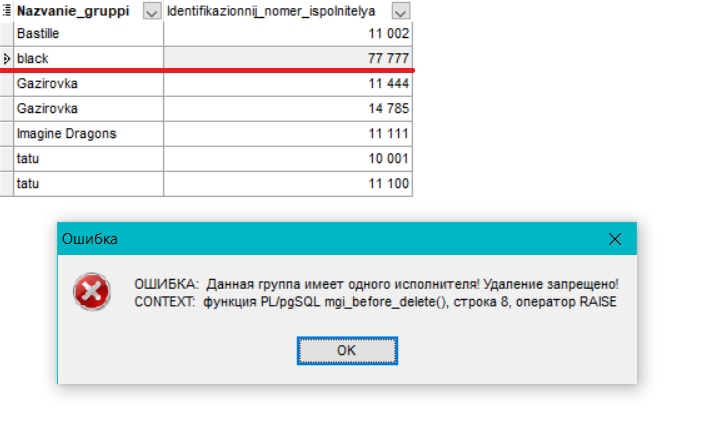


Рисунок 17 — сообщение об ошибке при удалении единственного исполнителя группы

Согласно бизнес-правилу предметной области «Музыкальные группы», каждая музыкальная группа должна содержать хотя бы одну песню.

Создадим триггер, который запрещает обновление внешнего ключа, если данная песня является единственной для этой группы :

CREATE FUNCTION pesnya\_before\_update() RETURNS trigger LANGUAGE plpgsql AS $$

DECLARE p integer;

BEGIN

SELECT COUNT(OLD."Nazvanie\_gruppi")

FROM "Pesnya" into p

WHERE "Nazvanie\_gruppi" =OLD."Nazvanie\_gruppi";

IF (p=1)THEN raise exception 'Данная песня является единственной для этой группы! ОБНОВЛЕНИЕ запрещено!';

END IF;

return OLD;

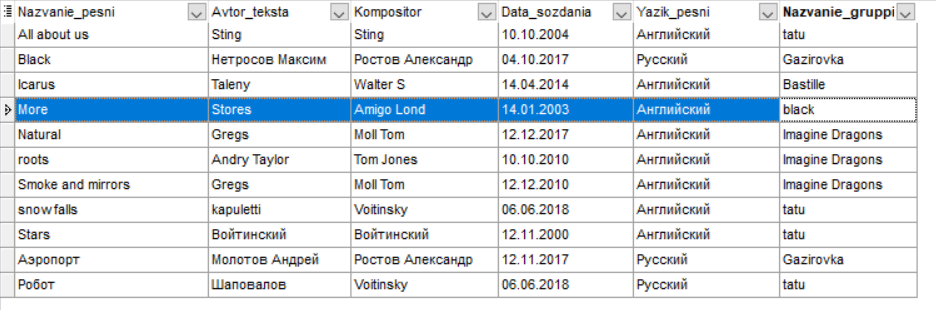
END;

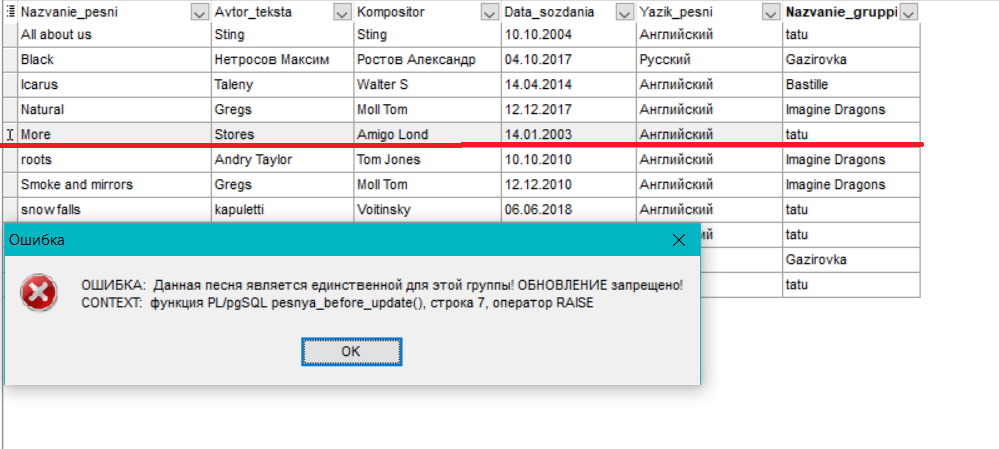
$$;

CREATE TRIGGER pesnya\_update

BEFORE UPDATE ON "Pesnya" FOR EACH ROW

EXECUTE PROCEDURE pesnya\_before\_update()

Рисунок 18 — песня More является единственной у группы black

Рисунок 19 — сообщение об ошибке при обновлении внешнего ключа

Создадим триггер, который запрещает удаление песни, если она является единственной для этой группы :

CREATE FUNCTION pesnya\_before\_delete() RETURNS trigger LANGUAGE plpgsql AS $$

DECLARE k integer;

BEGIN

SELECT COUNT(OLD."Nazvanie\_gruppi")

FROM "Pesnya" into k

WHERE "Nazvanie\_gruppi" =OLD."Nazvanie\_gruppi";

IF (k=1)THEN raise exception 'Данная песня является единственной для этой группы! Удаление запрещено!';

END IF;

return OLD;

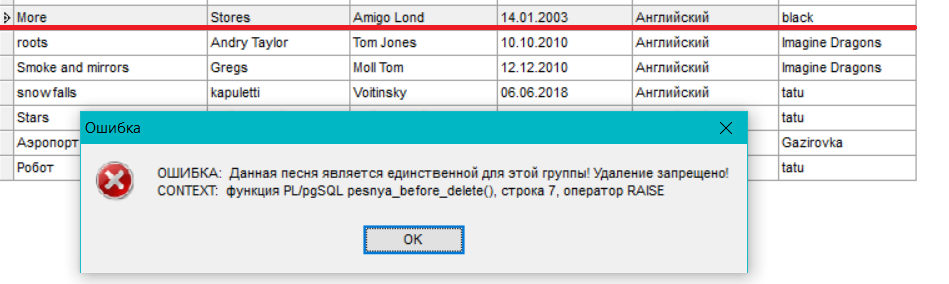
END;

$$;

CREATE TRIGGER pesnya\_delete

BEFORE DELETE ON "Pesnya" FOR EACH ROW

EXECUTE PROCEDURE pesnya\_before\_delete()

Рисунок 20 — сообщение об ошибке при удалении единственной песни у группы

Создадим триггер, который запрещает удалять единственное выступление в гастролях :

CREATE FUNCTION vistuplenie\_before\_delete() RETURNS trigger LANGUAGE plpgsql AS $$

DECLARE k integer;

BEGIN

SELECT COUNT(OLD."Nazvanie")

FROM "Vistuplenie" into k

WHERE "Nazvanie" =OLD."Nazvanie";

IF (k=1)THEN raise exception 'Данное выступление является единственным в гастрольной программе этой группы! Удаление запрещено!';

END IF;

return OLD;

END;

$$;

CREATE TRIGGER vistuplenie\_delete

BEFORE DELETE ON "Vistuplenie" FOR EACH ROW

EXECUTE PROCEDURE vistuplenie\_before\_delete()

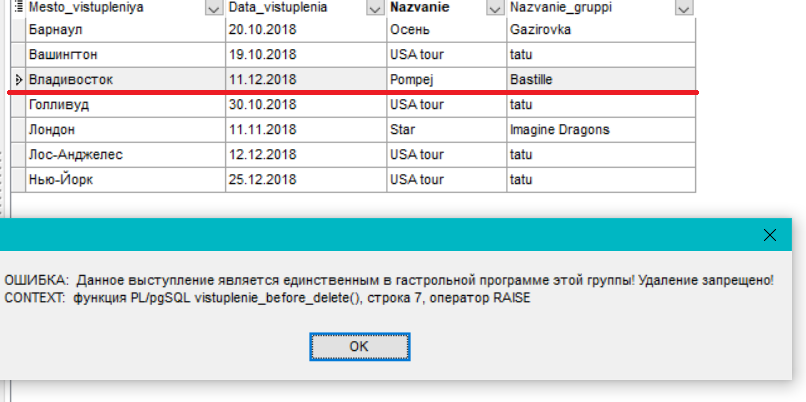


Рисунок 21 — сообщение об ошибке удаления единственного выступления

Создадим триггер, который запрещает обновление внешнего ключа , если выступление является единственным для этой группы :

CREATE FUNCTION vistuplenie\_before\_update() RETURNS trigger LANGUAGE plpgsql AS $$

DECLARE k integer;

BEGIN

SELECT COUNT(OLD."Nazvanie")

FROM "Vistuplenie" into k

WHERE "Nazvanie" =OLD."Nazvanie";

IF (k=1)THEN raise exception 'Данное выступление является единственным в гастрольной программе этой группы! ОБНОВЛЕНИЕ запрещено!';

END IF;

return OLD;

END;

$$;

CREATE TRIGGER vistuplenie\_update

BEFORE UPDATE ON "Vistuplenie" FOR EACH ROW

EXECUTE PROCEDURE vistuplenie\_before\_update()

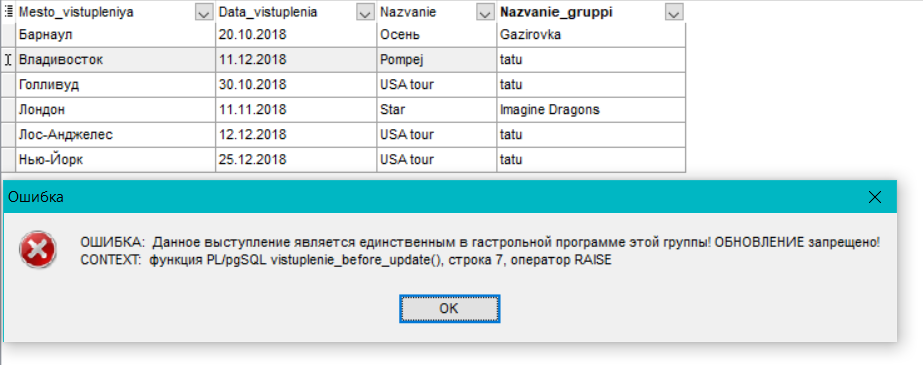


Рисунок 22 — сообщение об ошибке при обновлении внешнего ключа

Согласно бизнес-правилу предметной области «Музыкальные группы», музыкальная группа обязательно должна иметь, как минимум, одну песню. Поэтому при создании новой музыкальной группы, создадим для нее по умолчанию песню в таблице «Песня».

Создадим триггер для события вставки в таблицу «Музыкальная группа» :

CREATE FUNCTION musik\_group\_insert() RETURNS trigger LANGUAGE plpgsql AS $$

BEGIN

insert into "Pesnya" ("Nazvanie\_pesni", "Avtor\_teksta", "Kompositor", "Data\_sozdania", "Yazik\_pesni", "Nazvanie\_gruppi") values ('Новая песня', 'Новый автор текста', 'Новый композитор', null, null, NEW."Nazvanie\_gruppi");

RETURN NULL;

END;

$$;

CREATE TRIGGER musik\_gruppa\_after\_insert

AFTER INSERT ON "Musikalnaya\_gruppa" FOR EACH ROW

EXECUTE PROCEDURE musik\_group\_insert()

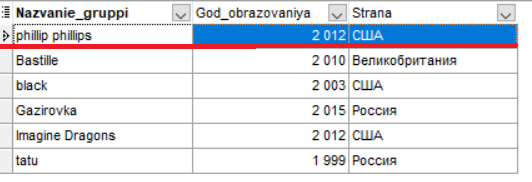
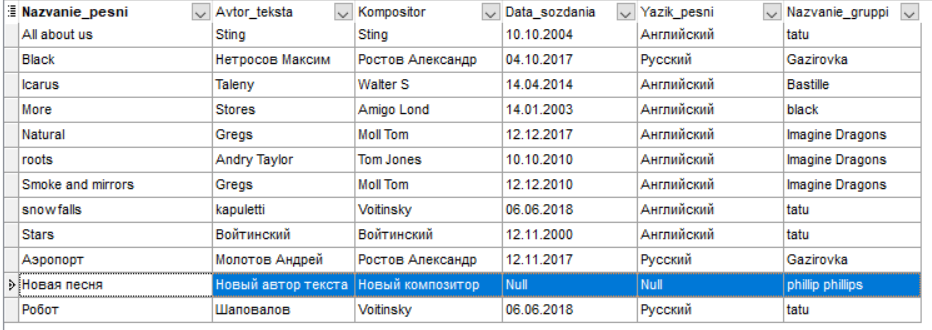


Рисунок 23 — создание новой музыкальной группы

Рисунок 24 — созданная по умолчанию для этой группы песня

Исходя из бизнес-правила предметной области «Музыкальные группы», хит-парад может содержать не более 100 мест. Создадим триггер, который будет проверять при добавлении или обновлении номера в хит-параде :

CREATE FUNCTION mesto\_hit\_parad() RETURNS trigger LANGUAGE plpgsql AS $$

BEGIN

IF (NEW."Nomer\_v\_hit\_parade">100)

THEN raise exception 'В хит-параде всего 100 мест!';

END IF;

RETURN NULL;

END;

$$;

CREATE TRIGGER hit\_parad

AFTER INSERT ON "Mesto\_v\_poslednem\_hit\_parade" FOR EACH ROW

EXECUTE PROCEDURE mesto\_hit\_parad()

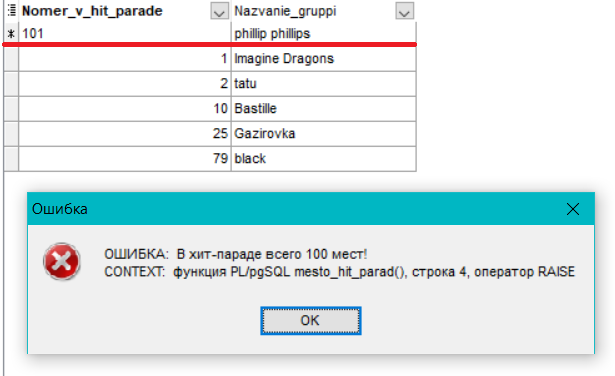


Рисунок 25 — сообщение об ошибке при добавлении номера >100

Бизнес-правило предметной области : цена билета на концерт группы зависит от номера в хит-параде и места выступления. Если группа находится в ТОП-5 хит-парада, то цена билета должна быть не менее 10000. Если место выступления группы является город Чикаго или Лондон, или Москва — не менее 15000, так как в данных городах билеты на концерт, в среднем, являются самыми дорогими.

Триггер на событие INSERT в таблице «Цена» :

CREATE FUNCTION Formirovanie\_zeni() RETURNS trigger LANGUAGE plpgsql AS $$

BEGIN

IF (NEW."Nomer\_v\_hit\_parade"<=5 and NEW."Zena\_bileta"<10000)

THEN raise exception 'Данная группа входит в ТОП-5 хит-парада! Цена билета должна быть не менее 10000!';

END IF;

IF (NEW."Mesto\_vistupleniya"='Лондон' or NEW."Mesto\_vistupleniya"='Чикаго' or NEW."Mesto\_vistupleniya"='Москва' and NEW."Zena\_bileta"<15000)

THEN raise exception 'Данная группа выступает в самом дорогом городе! Цена билета должна быть не менее 15000!';

END IF;

RETURN NULL;

END;

$$;

CREATE TRIGGER ZENA

AFTER INSERT ON "Zena" FOR EACH ROW

EXECUTE PROCEDURE Formirovanie\_zeni()

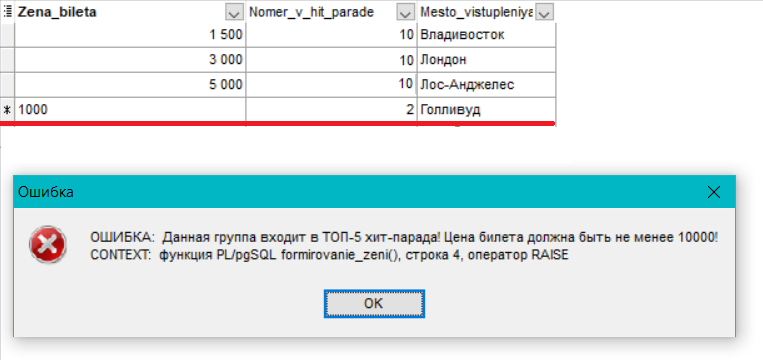
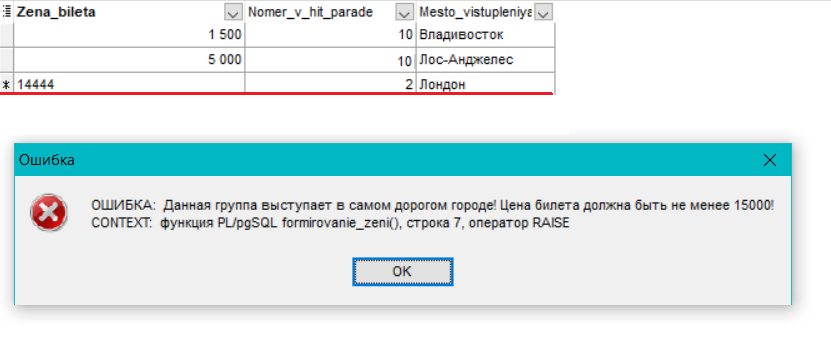


Рисунок 26 - добавляем цену билета <10000 для группы, которая входит в ТОП-5 хит-парада

Рисунок 27 — добавляем цену для группы, которая выступает в самом дорогом городе