ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Факультет 1

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №** 4

по дисциплине Системы управления базами данных

|  |  |  |
| --- | --- | --- |
| **ВЫПОЛНИЛ** | | |
| студент | РПИС-11  (группа) | Яковлев П. Н.  (ФИО) |
|  | **ПРОВЕРИЛ** |  |
| (должность) | | Борисов В.В.  (ФИО) |

Самара

2023

**Цель работы:**

* Знакомство с языком PL/pgSQL
* Создание процедур, функций и триггеров, обеспечивающих удобство работы с базой и обеспечение целостности
* Использование транзакция для обеспечения целостности

imdb=# create extension fuzzystrmatch;

CREATE OR REPLACE FUNCTION add\_movie(title VARCHAR(70), production\_year INTEGER)

RETURNS void AS $$

DECLARE phonetic\_code VARCHAR(70);

BEGIN

SELECT soundex(title) INTO phonetic\_code;

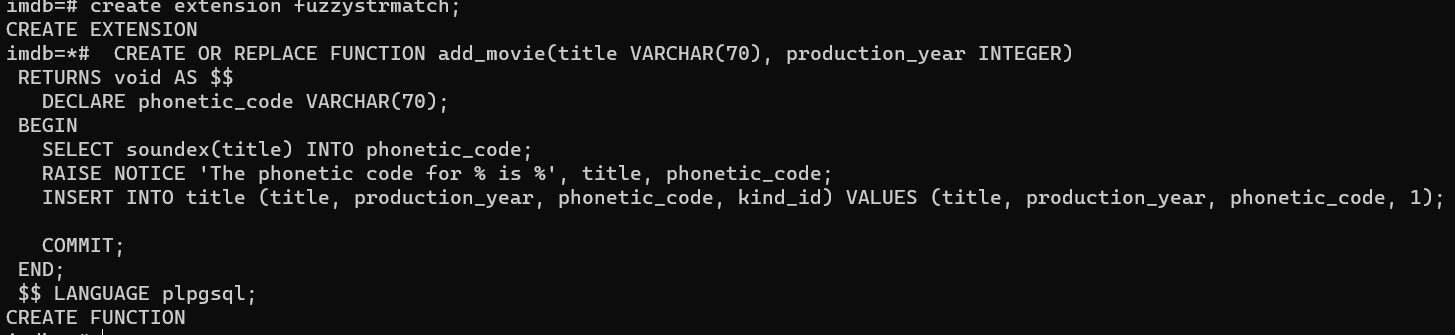
RAISE NOTICE 'The phonetic code for % is %', title, phonetic\_code;

INSERT INTO title (title, production\_year, phonetic\_code, kind\_id) VALUES (title, production\_year, phonetic\_code, 1);

COMMIT;

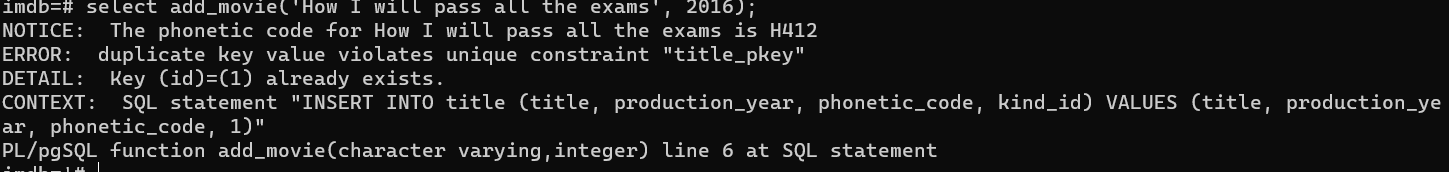
END;

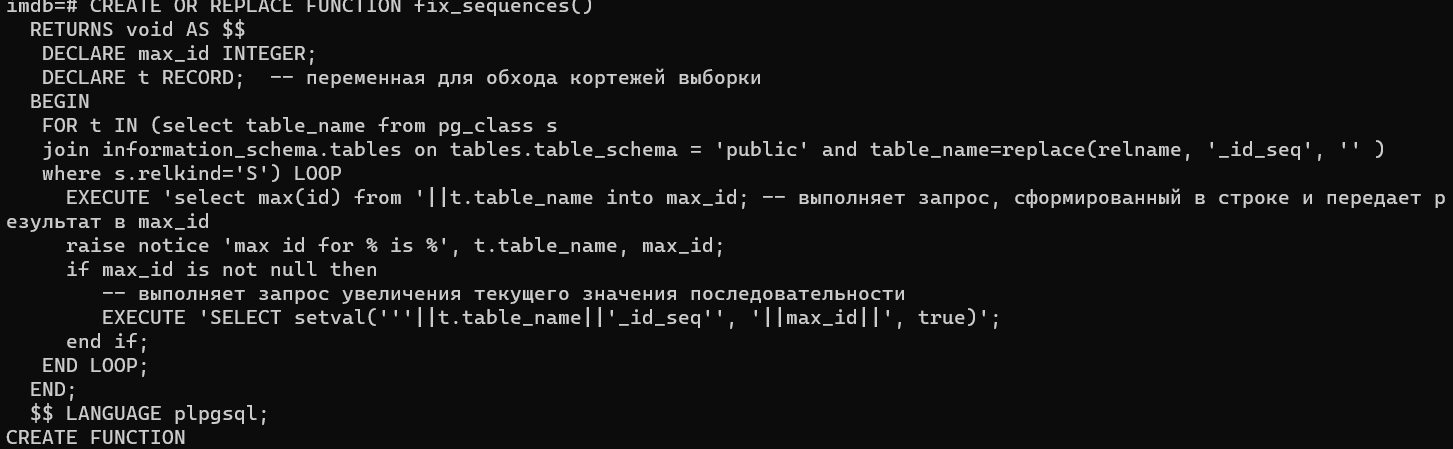
$$ LANGUAGE plpgsql;



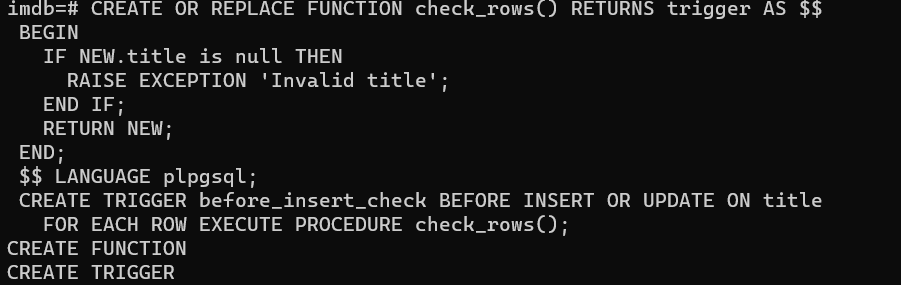
select add\_movie('How I will pass all the exams', 2016);

Ожидаемое сообщение об ошибке

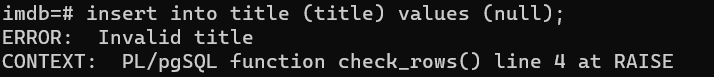




Триггеры



Триггер не дает добавлять строки в таблицу title, если поле title пусто:



Транзакции

BEGIN; -- начало транзакции

-- выполнение операций

SAVEPOINT my\_savepoint; -- сохранить точку, на которую можно потом вернуться,

-- но изменения в базу все еще не вносятся

-- выполнение операций

ROLLBACK TO my\_savepoint; -- после сохранения произошла ошибка, откат на состояние my\_savepoint

-- выполнение другого сценария

IF check\_ok() THEN:

COMMIT; -- фиксация результата

ELSE

ROLLBACK;

END IF;

Задания лабораторной работы

1. Сделать таблицу с историей изменений company\_name, в которую при обновлении через триггер записываются прежние значения и дата окончания их действий (дата обновления). Сделайте запрос, который показывает значения для конкретного кортежа на заданный момент времени.

2. Создать функцию с входным параметром - имя актера. Если актер с таким именем не найден, функция должна вернуть 0. Если найден, то вычислить через person\_info birth date (info\_type\_id=21) и death date (info\_type\_id=23) возраст актера. Если актер еще не умер, то вычислить, сколько сейчас лет. Функция должна вернуть целое число прожитых лет.

3. Используя функцию, созданную ранее, создайте процедуру, которая по имени актера выводит в лог текст следующего содержания:

Name: ...

Nicknames: nickname1, nickname2, ... строка выводится только если у человека есть клички, выводятся все клички через запятую

Age: ... вычисляется из функции

First appear: ... фильм и год выхода, в котором актер впервые снялся.

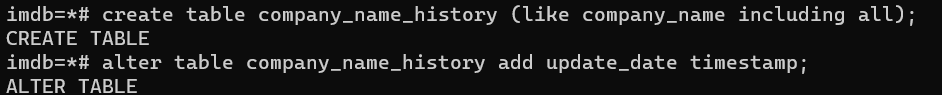
Если актер не найден или не удалось определить его первую роль, то процедура должна завершиться с ошибкой и вывести в лог "Invalid data"

1. Сделать таблицу с историей изменений company\_name, в которую при обновлении через триггер записываются прежние значения и дата окончания их действий (дата обновления). Сделайте запрос, который показывает значения для конкретного кортежа на заданный момент времени.

Создание новой таблицы и добавление в нее поля updated\_date, который будет хранить дату, и время, когда произошло изменение:

create table company\_name\_history (like company\_name including all);

alter table company\_name\_history add update\_date timestamp;



Создание функции для триггера:

CREATE OR REPLACE FUNCTION write\_company\_name\_history() RETURNS trigger AS $$

Begin

if (TG\_OP = 'UPDATE') then

insert into company\_name\_history (id, name, country\_code, imdb\_id, name\_pcode\_nf, name\_pcode\_sf, md5sum, update\_date)

values (old.id, old.name, old.country\_code, old.imdb\_id, old.name\_pcode\_nf, old.name\_pcode\_sf, old.md5sum, now());

end if;

return new;

End;

$$ language plpgsql;

Создание триггера:

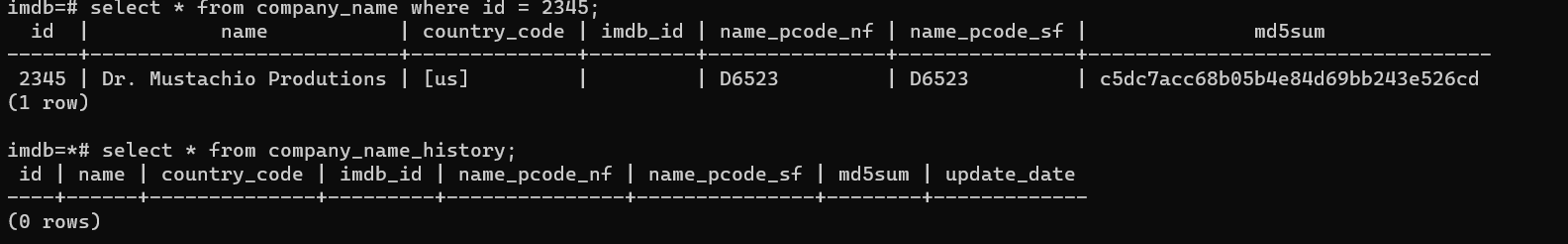
CREATE TRIGGER company\_name\_history

BEFORE UPDATE ON company\_name

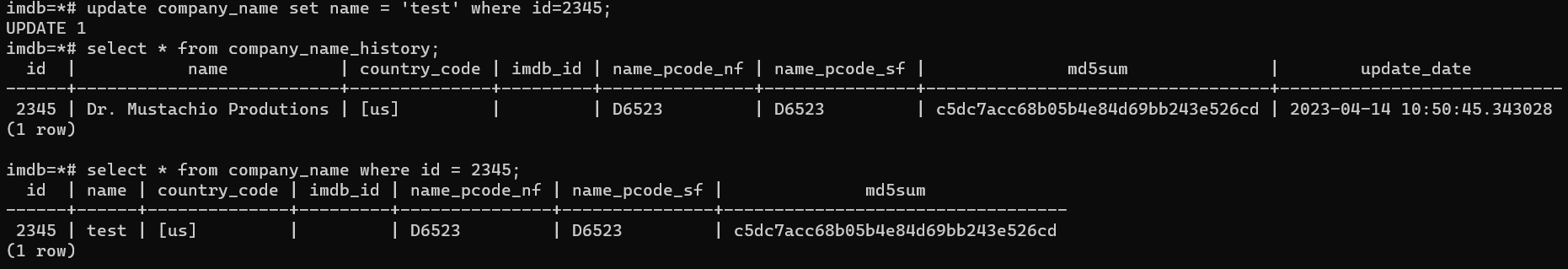
FOR EACH ROW

EXECUTE FUNCTION write\_company\_name\_history();

До изменения:



После:

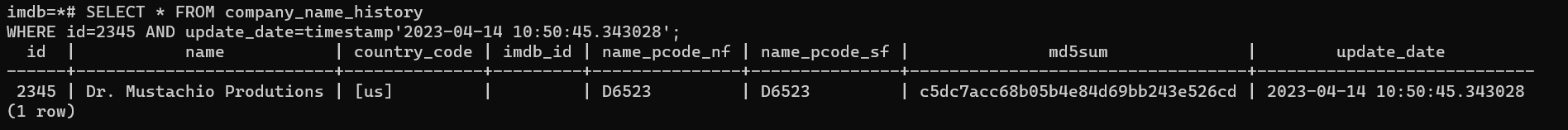


update company\_name set name = 'test' where id = 2345;

**запрос, который показывает значения для конкретного кортежа на заданный момент времени:**

SELECT \* FROM company\_name\_history

WHERE id=2345 AND update\_date=timestamp'2023-04-14 10:50:45.343028';



2. Создать функцию с входным параметром - имя актера. Если актер с таким именем не найден, функция должна вернуть 0. Если найден, то вычислить через person\_info birth date (info\_type\_id=21) и death date (info\_type\_id=23) возраст актера. Если актер еще не умер, то вычислить, сколько сейчас лет. Функция должна вернуть целое число прожитых лет.

CREATE OR REPLACE FUNCTION get\_actor\_age(name\_param TEXT)

RETURNS INTEGER AS $$

DECLARE

birth\_date DATE;

death\_date DATE;

age INTEGER;

BEGIN

SELECT info INTO birth\_date FROM person\_info WHERE person\_id = (SELECT id FROM name WHERE name = name\_param) AND info\_type\_id = 21;

IF birth\_date IS NULL THEN

RETURN 0;

END IF;

SELECT info INTO death\_date FROM person\_info WHERE person\_id = (SELECT id FROM name WHERE name = name\_param) AND info\_type\_id = 23;

IF death\_date IS NOT NULL THEN

age := EXTRACT(YEAR FROM death\_date) - EXTRACT(YEAR FROM birth\_date);

ELSE

age := EXTRACT(YEAR FROM CURRENT\_DATE) - EXTRACT(YEAR FROM birth\_date);

END IF;

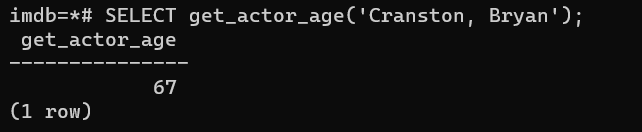
RETURN age;

END;

$$ LANGUAGE plpgsql;

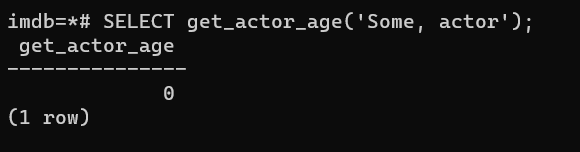
Пример вызова функции:

SELECT get\_actor\_age('Cranston, Bryan');



Если актер не найден:

SELECT get\_actor\_age('Some, actor');



3. Используя функцию, созданную ранее, создайте процедуру, которая по имени актера выводит в лог текст следующего содержания:

Name: ...

Nicknames: nickname1, nickname2, ... строка выводится только если у человека есть клички, выводятся все клички через запятую

Age: ... вычисляется из функции

First appear: ... фильм и год выхода, в котором актер впервые снялся.

Если актер не найден или не удалось определить его первую роль, то процедура должна завершиться с ошибкой и вывести в лог "Invalid data"

Процедура:

CREATE OR REPLACE PROCEDURE actor\_info(name\_in VARCHAR)

LANGUAGE plpgsql

AS $$

DECLARE

age INT;

first\_prodyear VARCHAR;

first\_movie VARCHAR;

nicknames VARCHAR;

BEGIN

SELECT get\_actor\_age(name\_in) INTO age;

IF age = 0 THEN

RAISE EXCEPTION 'Invalid data';

END IF;

SELECT ARRAY\_TO\_STRING(ARRAY\_AGG(aka\_name.name), ', ') INTO nicknames

FROM name

JOIN aka\_name ON name.id = aka\_name.person\_id

WHERE name.name = name\_in;

SELECT title INTO first\_movie

FROM title

JOIN cast\_info ON title.id = cast\_info.movie\_id

JOIN name ON cast\_info.person\_id = name.id

WHERE name.name = name\_in

ORDER BY title.production\_year

LIMIT 1;

SELECT production\_year INTO first\_prodyear

FROM title

JOIN cast\_info ON title.id = cast\_info.movie\_id

JOIN name ON cast\_info.person\_id = name.id

WHERE name.name = name\_in

ORDER BY title.production\_year

LIMIT 1;

IF NOT FOUND THEN

RAISE EXCEPTION 'Invalid data';

END IF;

RAISE NOTICE 'Name: %', name\_in;

IF nicknames IS NOT NULL THEN

RAISE NOTICE 'Nicknames: %', nicknames;

END IF;

RAISE NOTICE 'Age: %', age;

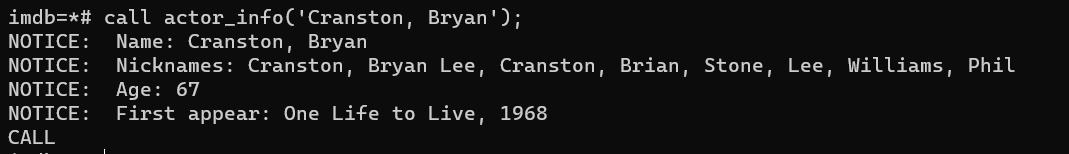
RAISE NOTICE 'First appear: %', first\_movie || ', ' || first\_prodyear;

END;

$$;

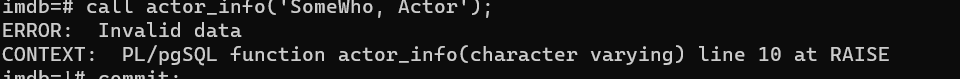
Вызов процедуры:

call actor\_info('Cranston, Bryan');



Вывод ошибки, если такого актера нет:

call actor\_info('SomeWho, Actor');



**Вывод:**

* Познакомились с языком PL/pgSQL
* Научились создавать процедуры, функции и триггеры, обеспечивающие удобство работы с базой и обеспечение целостности
* Научились использовать транзакция для обеспечения целостности