Manuel Rodríguez

FUNCTIONS, PROCEDURES, CURSORS, TRIGGERS...

1ºDAW DATABASE

INDEX

Function

This function is used to find all the films whose duration is higher than 160 minutes.

```
1  CREATE FUNCTION peliculas_largas()
2    RETURNS TABLE(title varchar, length smallint) AS
3  $BODY$
4  BEGIN
5    RETURN QUERY
6    SELECT film.title, film.length
7    FROM film
8    WHERE film.length > 160;
9  END;
10 $BODY$
11 LANGUAGE plpgsql;
12
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 48 msec.
```

After do a SELECT * from peliculas_largas(); this is the output.

	title character varying	length smallint
1	Agent Truman	169
2	Alley Evolution	180
3	Analyze Hoosiers	181
4	Anonymous Human	179
5	Antitrust Tomatoes	168
6	Artist Coldblooded	170
7	Dying Maker	168
8	Atlantis Cause	170
9	Badman Dawn	162
10	Baked Cleopatra	182
11	Casualties Encino	179

;

Stored procedure

This stored procedure allows us to increase the rental price of a film.

```
create or replace procedure increment_price (
    id integer,
    percentage integer
)
language plpgsql
as $$
begin
    update film
    set rental_rate = rental_rate * percentage/100
    where film_id = id;
    commit;

end;$$

```

For example, if I want to increase the price of the film with id 13 in a 20%, I should do that:

```
call increment_price(13,20);
```

Cursor

Here, I am going to create a cursor to obtain the title and release year of a film. I have done it in 2 different ways (loop while and loop for). For me it's better with a loop for because it's more simple as you can see in the code.

```
do $$
declare
    Rec_films Record;
    Cursor_films Cursor for select * from "film" order by film_id;
begin

Open Cursor_films;
Fetch Cursor_films into Rec_films;
While (FOUND) loop
Raise Notice 'Title: % || Release Year: % ||', Rec_films.title, Rec_films.release_year;
Fetch Cursor_films into Rec_films;
end loop;
end $$
language 'plpgsql';
```

```
do $$
declare
    Rec_films Record;
    Cursor_films Cursor for select * from "film" order by film_id;
begin

for Rec_films in Cursor_films loop
Raise Notice 'Title: % || Release Year: % ||', Rec_films.title, Rec_films.release_year;
end loop;
end $$
language 'plpgsql';
```

Here you can see the output:

```
NOTICE:  Title: Academy Dinosaur || Release Year: 2006 ||
NOTICE:  Title: Ace Goldfinger || Release Year: 2006 ||
NOTICE:  Title: Adaptation Holes || Release Year: 2006 ||
NOTICE:  Title: Affair Prejudice || Release Year: 2006 ||
NOTICE:  Title: African Egg || Release Year: 2006 ||
NOTICE:  Title: Agent Truman || Release Year: 2006 ||
NOTICE:  Title: Airplane Sierra || Release Year: 2006 ||
NOTICE:  Title: Airport Pollock || Release Year: 2006 ||
NOTICE:  Title: Alabama Devil || Release Year: 2006 ||
NOTICE:  Title: Aladdin Calendar || Release Year: 2006 ||
NOTICE:  Title: Alamo Videotape || Release Year: 2006 ||
NOTICE:  Title: Alaska Phantom || Release Year: 2006 ||
NOTICE:  Title: Ali Forever || Release Year: 2006 ||
NOTICE:  Title: Alice Fantasia || Release Year: 2006 ||
NOTICE:  Title: Alien Center || Release Year: 2006 ||
NOTICE:  Title: Alley Evolution || Release Year: 2006 ||
NOTICE:  Title: Alone Trip || Release Year: 2006 ||
```

Triggers

First of all, I have created a new table called old_films. My idea is to create a trigger so that when an update is executed on the 'film' table, the previous information before that update is saved.

```
create function Function_Trigger() returns Trigger
as
$$
begin
insert into "old_films" values(old.film_id, old.title, old.description, old.release_year,
return new;
end
$$
language plpgsql;

CREATE Trigger film_update before Update on film
for each row
execute procedure Function_Trigger();
```

So, now, let’s see if our trigger is working properly. I have executed this update:

```
15  UPDATE film set
16  title = 'Peli de Manu'
17  where title = 'Academy Dinosaur'
```

In my “film” table, the film has been updated successfully as you can see in the next picture:

	film_id [PK] integer	title character varying	description text
1	1	Peli de Manu	A Epic Drama of a Femir

But, let’s see if my trigger has been worked good and let’s see the table old_films:

	film_id integer	title character varying	description text
1	1	Academy Dinosaur	A Epic Drama of a Feminist And a Ma

And yes, as you can see, the film “Academy Dinosaur” has been saved in old_films before the update that I made.