

Ús de Bases de Dades

PRÀCTICA 2: Disseny de Bases de Dades i implementació de funcions i disparadors

Proposta de solució

Presentació

En aquesta Pràctica s'exerciten els aspectes que convé tenir en compte en el disseny d'una base de dades i la seva posterior implementació en un SGBD PostgreSQL.

Competències

En aquesta Pràctica es desenvolupen les següents competències del Grau en Multimèdia:

- Capacitat d'analitzar un problema en el nivell d'abstracció adequat a cada situació i aplicar les habilitats i coneixements adquirits per abordar-lo i resoldre'l.
- Capacitat per aplicar les tècniques específiques de tractament, emmagatzematge i administració de dades.

Objectius

Els objectius concrets d'aquesta Pràctica són:

- Saber fer el disseny lògic d'una base de dades relacional partint d'un disseny conceptual expressat amb el model ER.
- Conèixer el gestor de bases de dades relacionals amb suport per a objectes PostgreSQL.
- Saber utilitzar les sentències de l'SQL estàndard per a emprar servidors, catàlegs, esquemes, connexions, sessions i transaccions.
- Conèixer les diferències que hi ha entre la jerarquia de components de l'entorn SQL definida en l'SQL estàndard i la que ofereix PostgreSQL.
- Conèixer les sentències que ofereix PostgreSQL per a fer servir esquemes i bases de dades.
- Completar l'estudi dels components lògics d'una base de dades; és a dir, els procediments emmagatzemats i els disparadors.
- Utilitzar les sentències que ofereix PostgreSQL per a definir procediments emmagatzemats i disparadors.

Enunciat de la Pràctica:

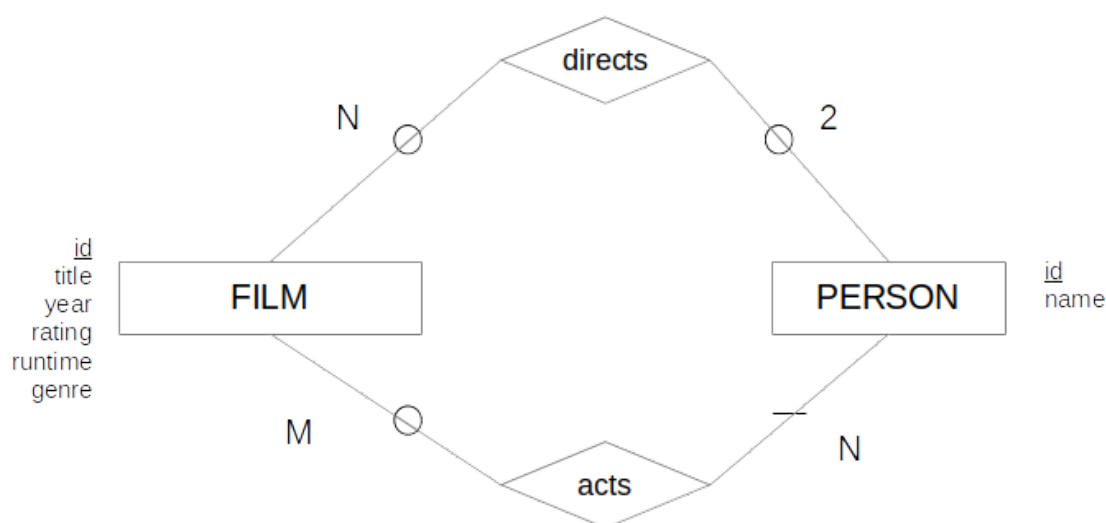
Es vol desenvolupar una base de dades que permeti gestionar les persones que han intervingut en diferents pel·lícules com actors i directors.

Ens proporcionen una taula, anomenada `pra2_raw`, sense cap clau primària i sense normalitzar. La descripció dels diferents camps d'aquesta taula és la següent:

- `film_id`: Identificador de la pel·lícula
- `title`: Títol de la pel·lícula
- `year`: Any d'estrena de la pel·lícula
- `genre`: Cadena de caràcters que defineix el gènere al que pertany la pel·lícula. De moment no ens interessa analitzar aquests valors
- `runtime`: Minuts de durada de la pel·lícula
- `rating`: Valora la conveniència de les pel·lícules per a certes audiències basant-se en el seu contingut segons el sistema de qualificacions de la *Motion Picture Association of America*
- `id_director`: Identificador de una de les persones que dirigeix la pel·lícula
- `director`: Nom del director al que correspon el valor `id_director` del mateix registre.
- `id_actor`: Identificador de una de les persones que actúa en la pel·lícula
- `actor`: Nom de l'actor al que correspon el valor `id_actor` del mateix registre.

La definició i dades d'aquesta taula corresponen al script `pra2_raw.sql` proporcionat.

Un cop analitzades les dades aportades s'ha creat el següent disseny conceptual per tal de ser implementat en un SGBD PostgreSQL:



Com podem veure en el diagrama, dues persones, com a màxim, poden intervenir com a directors d'una pel·lícula.

A partir d'aquesta descripció, cal respondre els següents exercicis de forma incremental.

En cada exercici cal descriure acuradament les tasques realitzades, el seu raonament i les comandes necessàries per l'adequada implementació.

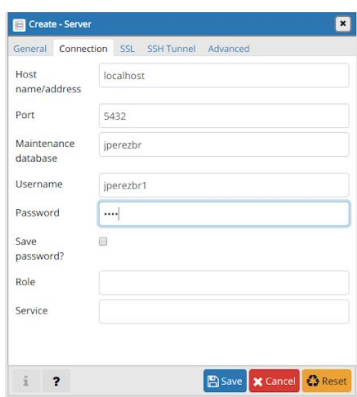
Pregunta 1 [5 %]

Crea l'esquema en la base de dades PostgreSQL (de nom *pra2data*) i un usuari (de nom *usuari1*, per exemple *jperezbr1*, amb el corresponent *password*) que tingui els permisos necessaris per crear i manipular les taules i les dades proporcionades. Les operacions posteriors, si no s'indica el contrari, caldrà realitzar-les amb l'usuari creat en aquest punt. Realitza la importació de la informació proporcionada en aquest esquema a partir del fitxer *pra2_raw.sql* proporcionat.

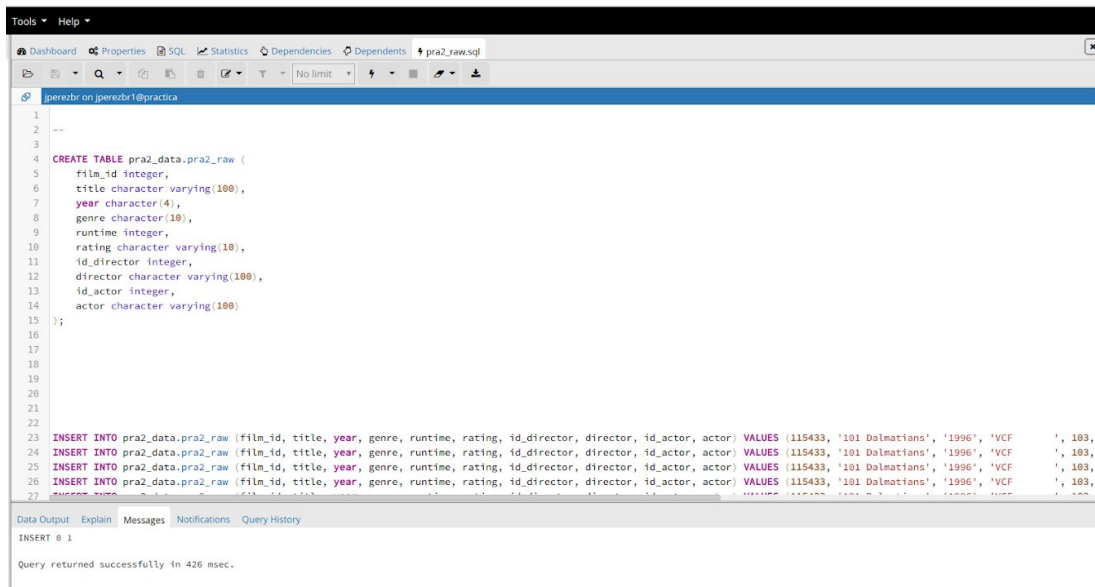
Proposta de solució

```
CREATE SCHEMA pra2data;
CREATE ROLE jperezbr1 LOGIN PASSWORD '1234';
GRANT ALL PRIVILEGES ON DATABASE jperezbr TO jperezbr1;
GRANT ALL PRIVILEGES ON SCHEMA pra2data TO jperezbr1;
```

Ens connectem amb el nou usuari,



I carreguem l'script proporcionat:



```

1  --
2
3
4  CREATE TABLE pra2_data.pra2_raw (
5      film_id integer,
6      title character varying(100),
7      year character(4),
8      genre character(10),
9      runtime integer,
10     rating character varying(10),
11     id_director integer,
12     director character varying(100),
13     id_actor integer,
14     actor character varying(100)
15 );
16
17
18
19
20
21
22
23 INSERT INTO pra2_data.pra2_raw (film_id, title, year, genre, runtime, rating, id_director, director, id_actor, actor) VALUES (115433, '101 Dalmatians', '1996', 'VCF', 103,
24 INSERT INTO pra2_data.pra2_raw (film_id, title, year, genre, runtime, rating, id_director, director, id_actor, actor) VALUES (115433, '101 Dalmatians', '1996', 'VCF', 103,
25 INSERT INTO pra2_data.pra2_raw (film_id, title, year, genre, runtime, rating, id_director, director, id_actor, actor) VALUES (115433, '101 Dalmatians', '1996', 'VCF', 103,
26
27

```

Data Output Explain Messages Notifications Query History

INSERT 0 1

Query returned successfully in 426 msec.

Pregunta 2 [15 %]

Implementar l'estructura definitiva de les taules, a partir del disseny proposat amb les corresponents claus primàries, claus alternatives i claus foranes. No es permet emprar noves claus artificials. Amb aquest objectiu creareu un segon esquema per aquesta pràctica (anomenat *pra2*) on l'*usuari1*, creat anteriorment, tindrà privilegis per a poder crear les taules.

Proposta de solució

Executem les següents sentències mitjançant la connexió realitzada amb l'usuari *jperezbr1*

```

CREATE SCHEMA pra2;

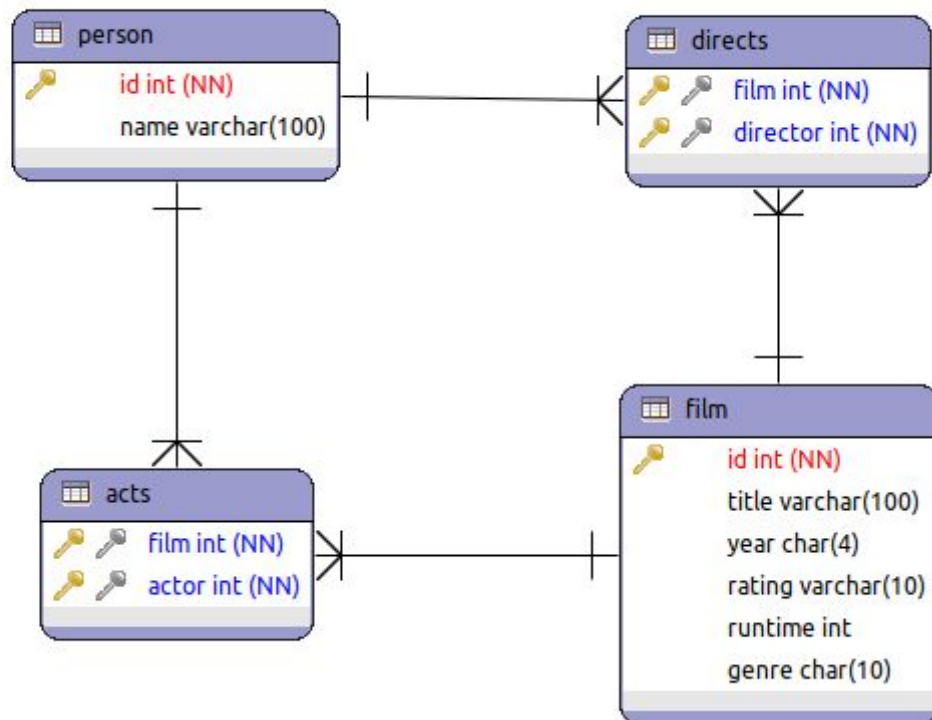
CREATE TABLE pra2.person(
    id integer,
    name varchar(100),
    CONSTRAINT person_pk PRIMARY KEY(id)
);

CREATE TABLE pra2.film(
    id integer,
    title varchar(100),
    year char(4),
    rating varchar(10),
    runtime integer,
    genre char(10),
    CONSTRAINT film_pk PRIMARY KEY(id)
);

```

```
CREATE TABLE pra2.directs(
    film integer,
    director integer,
    CONSTRAINT directs_pk PRIMARY KEY(film,director),
    CONSTRAINT directs2film_fk FOREIGN KEY (film ) REFERENCES pra2.film(id),
    CONSTRAINT directs2person_fk FOREIGN KEY (director) REFERENCES pra2.person(id)
);

CREATE TABLE pra2.acts(
    film integer,
    actor integer,
    CONSTRAINT acts_pk PRIMARY KEY(film,actor),
    CONSTRAINT acts2film_fk FOREIGN KEY (film ) REFERENCES pra2.film(id),
    CONSTRAINT acts2person_fk FOREIGN KEY (actor) REFERENCES pra2.person(id)
);
```



Pregunta 3 [25 %]

Crea el procés de càrrega de les dades auxiliars, les de la taula `pra2_raw` de l'esquema `pra2_data`, cap a les taules creades en el punt anterior, en l'esquema `pra2`. Es demana utilitzar quatre procediments emmagatzemats per a realitzar aquesta tasca.

```
LoadFilms()
LoadPersons()
LoadDirects()
LoadActs()
```

Aquests 4 procediments s'emmagatzemaran a l'esquema `pra2_data`.

En realitzar l'execució d'aquests procediments, en l'ordre adequat, totes les taules de l'esquema `pra2` emmagatzemaran els registres corresponents obtinguts des de la taula `pra2_data.pra2_raw`.

Proposta de solució

La implementació de les diferents funcions és la següent:

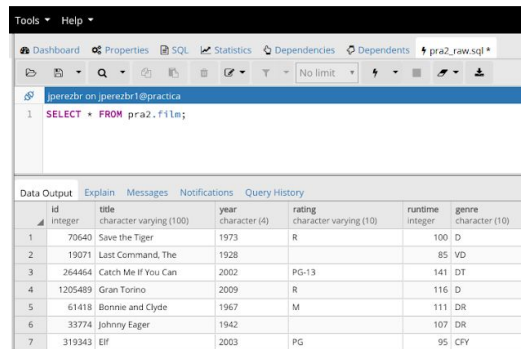
```
LoadFilms()
```

```
CREATE OR REPLACE FUNCTION pra2_data.LoadFilms() RETURNS void
AS $$
BEGIN
INSERT INTO pra2.film(id,title,year,rating,runtime,genre)
SELECT DISTINCT film_id,title,year,rating,runtime,genre FROM pra2_data.pra2_raw;
END;
$$LANGUAGE plpgsql;
```

Executem la funció i verifiquem el resultat

```
SELECT * FROM pra2_data.LoadFilms();
```

```
SELECT * FROM pra2.film;
```



Tools Help

Dashboard Properties SQL Statistics Dependencies Dependencies pra2_raw.sql

perzbr on perzbr1@practica

1 SELECT * FROM pra2_raw;file;

Data Output Explain Messages Notifications Query History

	id	title	year	rating	runtime	genre
	integer	character varying (100)	character (4)	character varying (10)	integer	character (10)
1	70640	Save the Tiger	1973	R	100	D
2	19071	Last Command, The	1928		85	VD
3	264464	Catch Me If You Can	2002	PG-13	141	DT
4	1205489	Gran Torino	2009	R	116	D
5	61418	Bonnie and Clyde	1967	M	111	DR
6	33774	Johnny Eager	1942		107	DR
7	319343	Elf	2003	PG	95	CFY

LoadPersons ()

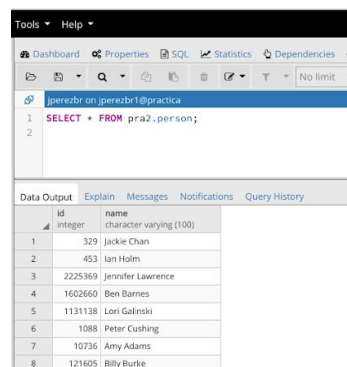
```
CREATE OR REPLACE FUNCTION pra2_data.LoadPersons() RETURNS void
AS $$
BEGIN
INSERT INTO pra2.person(id,name)
    SELECT DISTINCT id_director as id,director as name
    FROM pra2_data.pra2_raw WHERE id_actor IS NOT NULL
UNION
    SELECT DISTINCT id_actor as id,actor as name
    FROM pra2_data.pra2_raw
    WHERE id_actor IS NOT NULL;

END;
$$LANGUAGE plpgsql;
```

Executem la funció i verifiquem el resultat

```
SELECT * FROM pra2_data.LoadPersons();
```

```
SELECT * FROM pra2.person;
```



Tools Help

Dashboard Properties SQL Statistics Dependencies Dependencies

perzbr on perzbr1@practica

1 SELECT * FROM pra2.person;

2

Data Output Explain Messages Notifications Query History

	id	name
	integer	character varying (100)
1	329	Jackie Chan
2	453	Ian Holm
3	2225369	Jennifer Lawrence
4	1602660	Ben Barnes
5	1131138	Lori Loughlin
6	1088	Peter Cushing
7	10736	Amy Adams
8	121605	Billy Burke

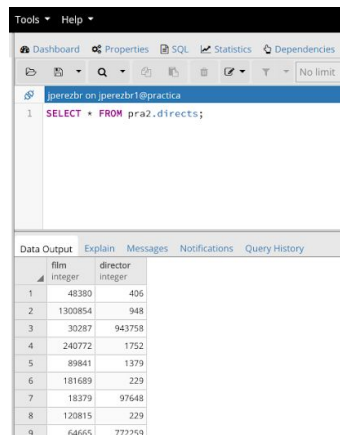
LoadDirects()

```
CREATE OR REPLACE FUNCTION pra2_data.LoadDirects() RETURNS void
AS $$
BEGIN
INSERT INTO pra2.directs(film,director)
        SELECT DISTINCT film_id as film,id_director as director
        FROM pra2_data.pra2_raw WHERE id_director IS NOT NULL;

END;
$$LANGUAGE plpgsql;

SELECT * FROM pra2_data.LoadDirects();

SELECT * FROM pra2.directs;
```



The screenshot shows a database IDE interface. The top menu bar includes 'Tools' and 'Help'. Below it, there are tabs for 'Dashboard', 'Properties', 'SQL', 'Statistics', and 'Dependencies'. The main editor area shows a SQL query: `SELECT * FROM pra2.directs;`. Below the editor, there is a 'Data Output' tab which displays the results of the query in a table format.

	film integer	director integer
1	48380	406
2	1300854	948
3	30287	943758
4	240772	1752
5	89841	1379
6	181689	229
7	18379	97648
8	120815	229
9	64665	772259

LoadActs()

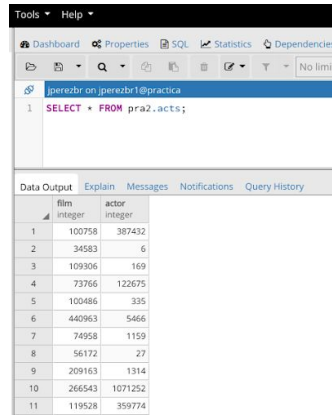
```
CREATE OR REPLACE FUNCTION pra2_data.LoadActs() RETURNS void
AS $$
BEGIN
INSERT INTO pra2.acts(film,actor)
        SELECT DISTINCT film_id as film,id_actor as actor
        FROM pra2_data.pra2_raw WHERE id_actor IS NOT NULL;

END;
$$LANGUAGE plpgsql;
```



```
SELECT * FROM pra2_data.LoadActs();
```

```
SELECT * FROM pra2.acts;
```



The screenshot shows a database query tool interface. At the top, there's a menu bar with 'Tools' and 'Help'. Below it, a toolbar contains icons for Dashboard, Properties, SQL, Statistics, and Dependencies. A search bar is present with the text 'jperesbr on jperesbr1@practica'. The main query editor displays the SQL statement: `SELECT * FROM pra2.acts;`. Below the editor, there's a 'Data Output' tab which is active, showing a table with 11 rows and 2 columns: 'film' (integer) and 'actor' (integer). The data is as follows:

	film integer	actor integer
1	100758	387432
2	34583	6
3	109306	169
4	73766	122675
5	100486	335
6	440963	5466
7	74958	1159
8	56172	27
9	209163	1314
10	266543	1071252
11	119528	359774

Pregunta 4 [25 %]

Crea una funció, ubicada a l'esquema *pra2*, amb el següent contracte **GetFilmsByDirector(director_name)**, que retorni un conjunt de registres amb els següents atributs:

title: Títol de la pel·lícula.

year: any d'estrena de la pel·lícula.

El resultat s'obté de la informació emmagatzemada en les taules de l'esquema *pra2*. Els registres resultants cal que estiguin ordenats de major a menor pel valor de l'atribut *year*.

En cas de que no existeixi informació de cap persona a partir del nom introduït cal que es mostri un missatge d'error. En cas que no es trobi cap pel·lícula dir, cal mostrar un missatge d'error diferent.

Exemplifica la funcionalitat de la funció desenvolupada.

Proposta de solució

```
CREATE TYPE pra2.type_film AS (
  title varchar (100),
  year integer
);

CREATE OR REPLACE FUNCTION pra2.GetFilmsByDirector(director_name pra2.person.name%type)
RETURNS SETOF pra2.type_film AS
$BODY$
DECLARE
  list pra2.type_film;
BEGIN
  IF ((SELECT COUNT(*) FROM pra2.person WHERE name=director_name)>0) THEN
    IF ((SELECT count(p.name)
      FROM pra2.person p, pra2.directs dir
      WHERE p.id=dir.director
      AND p.name like director_name )>=1)
    THEN
      FOR list IN SELECT f.title , f.year
        FROM pra2.person p, pra2.directs dir ,pra2.film f
        WHERE p.id=dir.director
```

```

        AND dir.film=f.id
        AND p.name like director_name
        ORDER BY f.year DESC
    LOOP
        RETURN NEXT list;
    END LOOP;

    ELSE RAISE EXCEPTION 'No films directed by %', director_name;
    END IF;

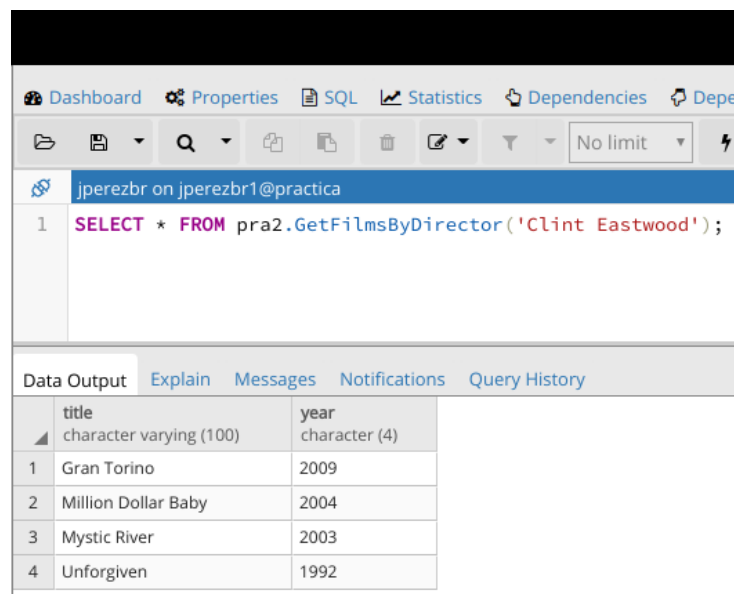
ELSE RAISE EXCEPTION 'Doesnt exists this person: %', director_name;
END IF;

EXCEPTION
WHEN raise_exception THEN
    RAISE EXCEPTION' %: %',SQLSTATE, SQLERRM;
WHEN others THEN
    RAISE EXCEPTION' P0001: internal error';
END;
$BODY$
LANGUAGE plpgsql;

```

Verifiquem la funcionalitat desenvolupada:

```
SELECT * FROM pra2.GetFilmsByDirector('Clint Eastwood');
```



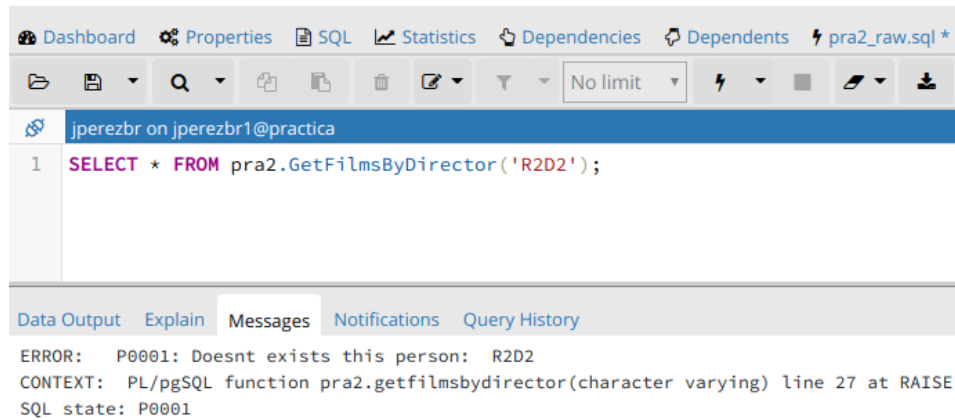
The screenshot shows a SQL IDE interface with a toolbar at the top containing icons for Dashboard, Properties, SQL, Statistics, Dependencies, and Depe. Below the toolbar is a search bar with the text 'jperezbr on jperezbr1@practica'. The main area displays a SQL query: `SELECT * FROM pra2.GetFilmsByDirector('Clint Eastwood');`. Below the query, there is a tabbed interface with 'Data Output' selected. The 'Data Output' tab shows a table with two columns: 'title' (character varying (100)) and 'year' (character (4)). The table contains four rows of data:

	title	year
1	Gran Torino	2009
2	Million Dollar Baby	2004
3	Mystic River	2003
4	Unforgiven	1992

I demostrem que les excepcions funcionen adequadament:

- En el cas de que el nom introduït no correspongui a cap persona.

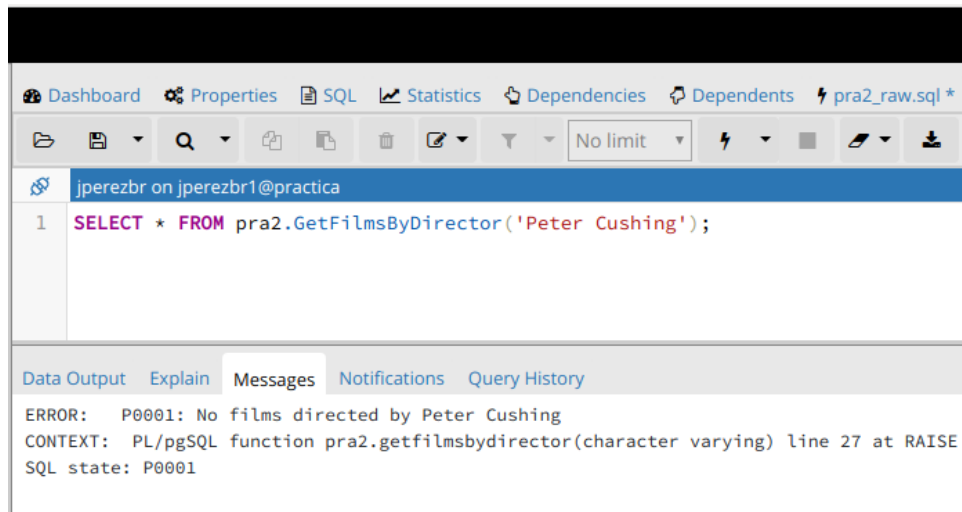
```
SELECT * FROM pra2.GetFilmsByDirector('R2D2');
```



The screenshot shows a SQL IDE interface with a toolbar at the top containing icons for Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. Below the toolbar, the user 'jperezbr' is logged in on 'jperezbr1@practica'. The SQL editor contains the query: `1 SELECT * FROM pra2.GetFilmsByDirector('R2D2');`. The 'Messages' tab is selected, displaying the following error:
ERROR: P0001: Doesnt exists this person: R2D2
CONTEXT: PL/pgSQL function pra2.getfilmsbydirector(character varying) line 27 at RAISE
SQL state: P0001

- En el cas de que el nom introduït correspongui a una persona però que no ha dirigit cap pel·lícula..

```
SELECT * FROM pra2.GetFilmsByDirector('Peter Cushing');
```



The screenshot shows the same SQL IDE interface. The SQL editor contains the query: `1 SELECT * FROM pra2.GetFilmsByDirector('Peter Cushing');`. The 'Messages' tab is selected, displaying the following error:
ERROR: P0001: No films directed by Peter Cushing
CONTEXT: PL/pgSQL function pra2.getfilmsbydirector(character varying) line 27 at RAISE
SQL state: P0001

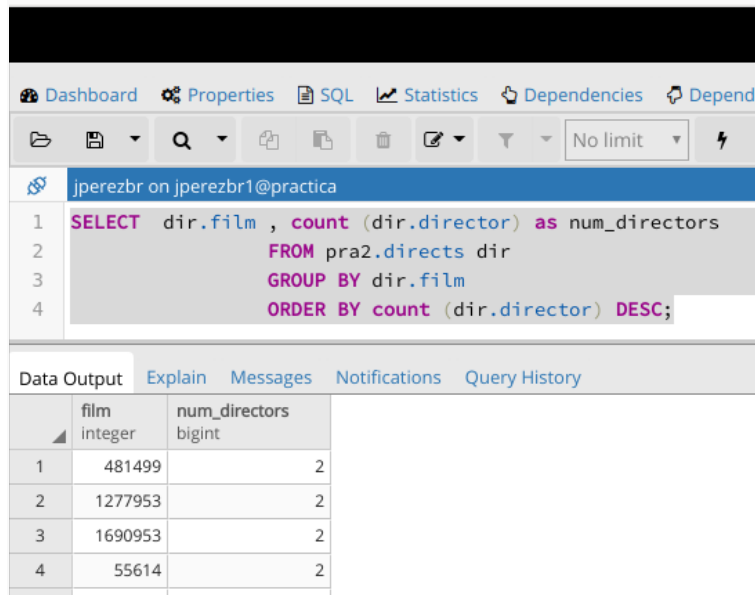
Pregunta 6 [25 %]

Implementar mitjançant disparadors la regla de negoci següent: "No pot ser que una pel·lícula hagi estat dirigida per més de dos persones".

Proposta de solució

Primer val la pena comprovar el nombre de persones que ha dirigit cada pel·lícula:

```
SELECT dir.film , count (dir.director) as num_directors
FROM pra2.directs dir
GROUP BY dir.film
ORDER BY count (dir.director) DESC;
```



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query is as follows:

```
1 SELECT dir.film , count (dir.director) as num_directors
2 FROM pra2.directs dir
3 GROUP BY dir.film
4 ORDER BY count (dir.director) DESC;
```

The results pane shows the following data:

	film integer	num_directors bigint
1	481499	2
2	1277953	2
3	1690953	2
4	55614	2
-	-----	-

Implementem la solució amb dos disparadors. Un primer disparador executat *BEFORE*, num_directorsxfilm_before, pot calcular el nombre de directors de cada pel·lícula abans de l'actualització o la inserció, i un segon disparador *AFTER*, num_directorsxfilm_after, pot verificar si es pot incloure una altra persona com a director d'una pel·lícula.

Aquesta solució té un problema. En PostgreSQL, els procediments invocats per a disparadors no poden retornar qualsevol valor; per tant, no tenim manera de retornar la quantitat d'empleats per a la nova oficina abans que es produeixi la modificació.

Per poder-ho fer, definirem la taula següent per tal que els procediments puguin obtenir el nombre de directors per pel·lícula actualitzats:

```
CREATE TABLE pra2.directors_x_film(
    film integer,
    num_directors integer
);
```

Ara ja estem en condicions d'implementar la solució. Primer implementem el disparador BEFORE i la funció associada.

```
CREATE OR REPLACE FUNCTION pra2.directors_before() RETURNS
TRIGGER AS $$
BEGIN
DELETE FROM pra2.directors_x_film;
INSERT INTO pra2.directors_x_film SELECT  dir.film , count (dir.director) as num_directors
    FROM pra2.directs dir
    GROUP BY dir.film
    ORDER BY count (dir.director) DESC;

RETURN NULL;
END
$$ LANGUAGE plpgsql;

CREATE TRIGGER num_directorsxfilm_before
BEFORE INSERT OR UPDATE ON pra2.directs
FOR EACH STATEMENT
EXECUTE PROCEDURE pra2.directors_before();
```

El disparador i la funció associada que s'executarà després de la inserció o modificació, tindrà el codi següent:

```
CREATE OR REPLACE FUNCTION pra2.directors_after() RETURNS
TRIGGER AS $$
DECLARE
num_directors_before integer DEFAULT 0;

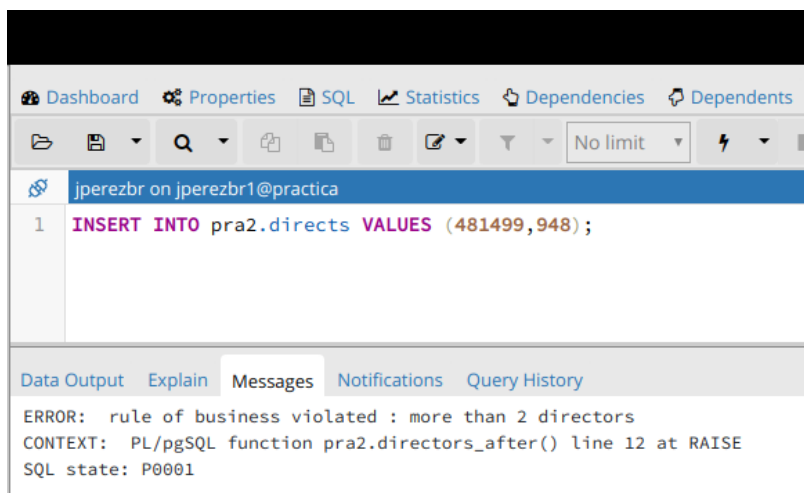
BEGIN
SELECT  num_directors INTO num_directors_before
    FROM pra2.directors_x_film
    WHERE film=NEW.film ;
```

```
RAISE NOTICE 'film with id=% had % directors',NEW.film,num_directors_before ;
IF (num_directors_before>=2) THEN
    RAISE EXCEPTION 'rule of business violated : more than 2 directors ';
END IF;
RETURN NULL;
END
$$ LANGUAGE plpgsql;
```

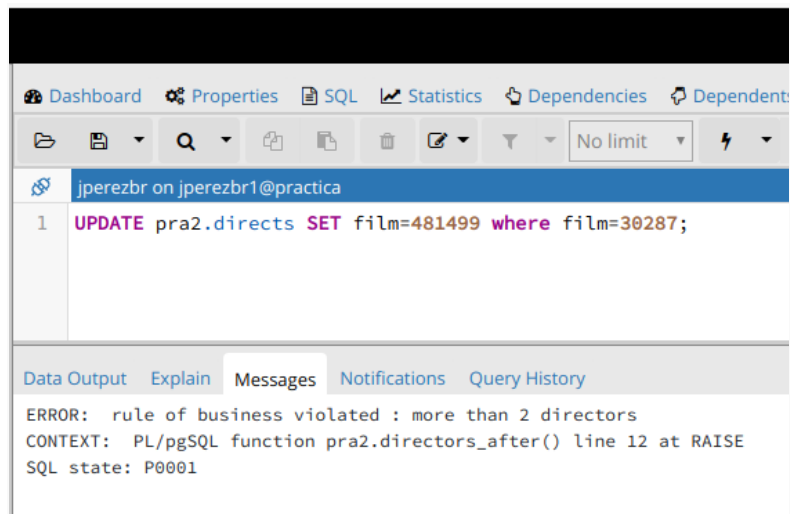
```
CREATE TRIGGER num_directorsxfilm_after
after INSERT OR UPDATE ON pra2.directs
FOR EACH row
EXECUTE PROCEDURE pra2.directors_after();
```

Podem verificar el funcionament de la regla de negoci implementada.

```
INSERT INTO pra2.directs VALUES (481499,948);
```



```
UPDATE pra2.directs SET film=481499 where film=30287;
```



```

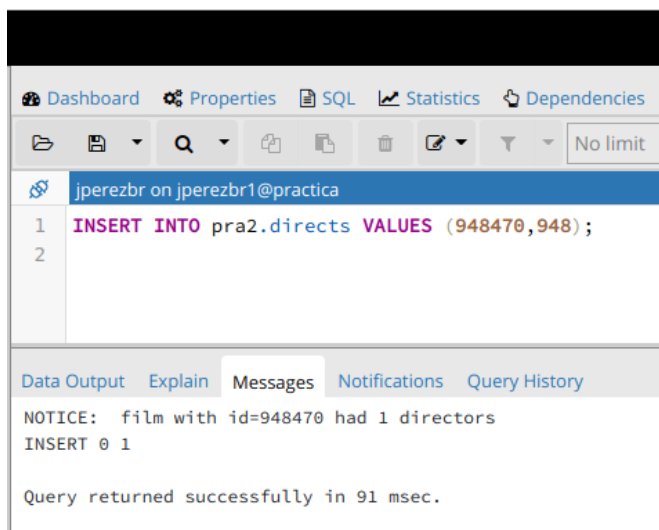
Dashboard Properties SQL Statistics Dependencies Dependent
jperzbr on jperzbr1@practica
1 UPDATE pra2.directs SET film=481499 where film=30287;

Data Output Explain Messages Notifications Query History
ERROR: rule of business violated : more than 2 directors
CONTEXT: PL/pgSQL function pra2.directors_after() line 12 at RAISE
SQL state: P0001

```

Fixeu-vos que, si la sentència de modificació no compleix la regla de negoci, el procediment emmagatzemat executat AFTER genera una excepció, la qual desfà la sentència que dispara el disparador i tota la transacció en curs. En canvi, si l'execució d'una sentència de modificació no provoca la violació de la regla de negoci, la sentència INSERT o UPDATE s'executa correctament.

```
INSERT INTO pra2.directs VALUES (948470,948);
```



```

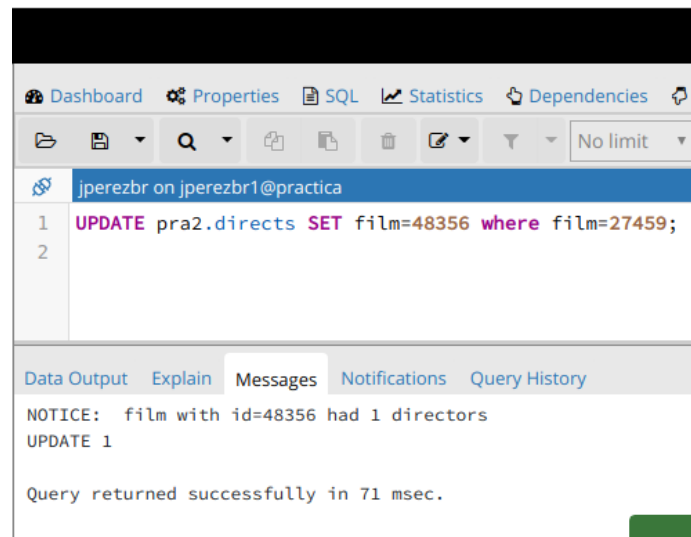
Dashboard Properties SQL Statistics Dependencies
jperzbr on jperzbr1@practica
1 INSERT INTO pra2.directs VALUES (948470,948);
2

Data Output Explain Messages Notifications Query History
NOTICE: film with id=948470 had 1 directors
INSERT 0 1

Query returned successfully in 91 msec.

```

```
UPDATE pra2.directs SET film=48356 where film=27459;
```

The screenshot shows a web-based SQL editor interface. At the top, there's a navigation bar with tabs: Dashboard, Properties, SQL, Statistics, and Dependencies. Below this is a toolbar with icons for file operations and a search bar. The main editor area shows a query being executed by user 'jperezbr' on 'jperezbr1@practica'. The query is:

```
1 UPDATE pra2.directs SET film=48356 where film=27459;
2
```

Below the query editor, there are tabs for Data Output, Explain, Messages, Notifications, and Query History. The 'Messages' tab is active, displaying the following output:

```
NOTICE: film with id=48356 had 1 directors
UPDATE 1

Query returned successfully in 71 msec.
```

Recursos

Els següents recursos són d'utilitat per la realització de la Pràctica:

Bàsics

- Document "Bases de dades en PostgreSQL"
- Document "El llenguatge SQL II"

Complementaris

- Document "Disseny de bases de dades".
- <https://www.postgresql.org/docs/10/static/index.html>
- http://cv.uoc.edu/app/mediawiki64/wiki/Exercicis_amb_PostgreSQL

Criteris d'avaluació

A continuació podeu consultar el pes de cadascun dels exercicis proposats en la nota final de la pràctica:

Exercici 1	5%
Exercici 2	15%
Exercici 3	25%
Exercici 4	25%
Exercici 5	30%

El lliurament s'haurà de realitzar en l'apartat de lliuraments en un fitxer de text (pdf o similar) que contingui:

- Les respostes a les preguntes plantejades.
- Les sentències en SQL plantejades i les respostes obtingudes.
- La justificació i criteris de disseny utilitzats en cada resposta.

IMPORTANT: Addicionalment caldrà implementar la pràctica al servidor <https://ubd.eimt.uoc.edu/eimtbdl/>. Cada alumne disposarà d'un usuari, que es proporcionarà individualment. És necessari que prèviament es realitzin les proves en local, a la instal·lació feta per vosaltres. A mesura que es vagin completant les tasques encomanades i ja es considerin definitives, es recomana anar implementant les instruccions corresponents a la base de dades que se us proporcionarà en aquest servidor.

No es considerarà vàlid el codi que no es trobi implementat en el servidor i justificat a la memòria de la pràctica.

Notes

Aquesta PRA s'ha de fer de manera estrictament individual. Qualsevol indicati de còpia serà penalitzat amb un suspens (D) per a totes les parts implicades i la possible avaluació negativa de l'assignatura en la seva totalitat.

Cal que l'estudiant citi totes les fonts que ha utilitzat per a la realització de la PRA. Si no és així, es considerarà que l'estudiant ha comès plagi, sent penalitzat amb un suspens (D) i la possible avaluació negativa de l'assignatura en la seva totalitat.

La data límit per al lliurament d'aquesta PRA és el dia 20 de maig de 2019.

La data de lliurament d'aquesta PRA ha de ser estrictament respectada, i no s'acceptarà cap lliurament després de la data establerta. Si es considera per alguna raó justificada que no es va a poder complir amb aquesta data, l'estudiant s'haurà de posar en contacte amb el seu consultor de l'assignatura **AMB SUFICIENT ANTERIORITAT** per poder buscar conjuntament una solució al respecte. Si s'acorda el lliurament amb posterioritat, la nota màxima d'aquesta PRA serà un aprovat (C+).