DATA STRUCTURE- First assignment

S. Xiaofan Fernández and Samai García (1292)

**2. Explore the database**

In the following lines we describe the different tables we have and the primary and foreign key for each one of them, in the format previously required:

actor (**actor\_id**, first\_name, last\_name, last\_uppdate)

address (**address\_id**, address, address2, district, city\_id → city.city\_id, postal\_code, phone, last\_update)

category (**category\_id**, name,last\_update)

city (**city\_id**, city, country\_id → country.country\_id, last\_update)

country (**country\_id**, country, last\_update)

customer (**customer\_id**,store\_id*,* first\_name, last\_name, email, address\_id → address.address\_id, activebool, create\_date, last\_update)

film (**film\_id**, title, description, release\_year, language\_id → language.language\_id, rental\_duration, rental\_rate, length, replacement\_cost, rating mpaa\_rating, last\_update, special\_features, fulltext)

film\_actor (**actor\_id** → actor.actor\_id, **film\_id** → film.film\_id, last\_update)

film\_category (**film\_id** → film.film\_id, **category\_id** → category.category\_id , last\_update)

inventory (**inventory\_id**, film\_id → film.film\_id, store\_id, last\_update)

language (**language\_id**, name, last\_update)

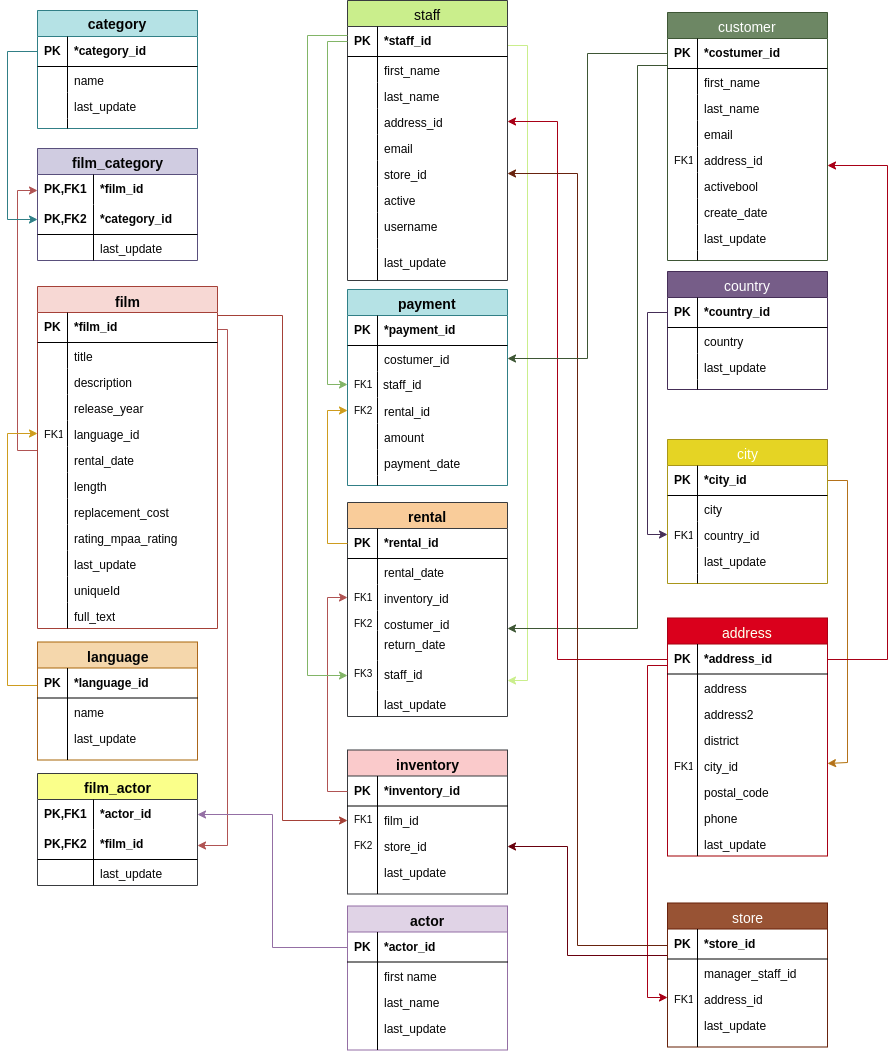
payment (**payment\_id**, customer\_id → customer.customer\_id, staff\_id → staff.staff\_id, rental\_id → rental.retnal\_id, amount, payment\_date)

rental (**rental\_id,** rental\_date, inventory-id → inventory.inventory\_id, customer\_id → customer.customer\_id, return\_date, staff\_id → staff.staff\_id, last\_update)

staff (**staff\_id**, first\_name, last\_name, address\_id → address.address\_id, email, store\_id, active, username, last\_update)

store (**store\_id**, manager\_staff\_id → staff.staff\_id, address\_id, last\_update)

Then, we created the corresponding relational schema:



**3. Queries**

In this part we were required to implement some queries on the database using SQL. All of them have been passed through a SQL formatter so that they are readable (http://www.dpriver.com/pp/ sqlformat.htm).

1. Number of movies rented each year. The query must return two attributes containing the year and the number of films rented. Order the answer by the rental year in ascending order.

SELECT *Count*(\*) AS number\_rented\_films,

**Extract** (year FROM rental\_date) AS year *--PICKS THE YEAR OF THE DATE*

FROM rental

GROUP BY year

ORDER BY ( year ) ASC *--ORDERS THE YEAR ASC. AND COUNTS THE NUMBER OF FILMS RENTED EACH OF THEM*

1. Client who has rented more movies. If there is a tie between several clients, all clients with the maximum number of rented movies must appear in the response. The query must return the attributes: customer id, first name, last name and the number of rented movies.

SELECT first\_name,

last\_name,

customer.customer\_id,

count

FROM customer,

(SELECT customer\_id,

count

FROM (SELECT customer\_id,

*Count*(\*)

FROM rental

GROUP BY customer\_id

ORDER BY *Count* (\*) DESC) AS customer

*-- LIST OF HOW MANY MOVIES THE CUSTOMERS RENTED*

WHERE count IN (SELECT *Max*(count)

FROM (SELECT customer\_id,

*Count*(\*)

FROM rental

GROUP BY customer\_id

ORDER BY *Count* (\*) DESC) AS x)) AS id\_client

*--X = CUSTOMER; ID\_CLIENT SELECTS THE TOP ONE*

WHERE id\_client.customer\_id = customer.customer\_id *-- CHOSE THE FIRST NAME, LAST NAME, ID AND COUNT*

1. List the cities where movies, in which “Bob Fawcett” appears, have been rented. Each city must appear just once. Sort the output alphabetically by the city name from A to Z. The query must return the city id and the city.name.

SELECT city.city\_id,

city.city

FROM city,

(SELECT address.address\_id,

address.city\_id

FROM address,

(SELECT store.store\_id,

store.address\_id

FROM store,

(SELECT inventory\_id,

inventory.store\_id

FROM inventory,

(SELECT film.film\_id

FROM film,

(SELECT film\_id

FROM film\_actor,

(SELECT actor\_id,

first\_name

FROM actor

WHERE first\_name = 'Bob'

AND last\_name = 'Fawcett'

) AS

actor\_bob

*-- SELECTS THE NAME BOB AND LAST NAME FAWCETT*

WHERE film\_actor.actor\_id =

actor\_bob.actor\_id) AS

film\_of\_bob

*--SELECTS ALL THE FILMS IN WHICH HE APPEARS*

WHERE film.film\_id = film\_of\_bob.film\_id) AS

inventory\_bob

*--SELECTS ALL THE MOVIES OF THE INVENTORY HE APPEARS*

WHERE inventory.film\_id = inventory\_bob.film\_id) AS

store\_bob

WHERE store\_bob.store\_id = store.store\_id) AS store\_id\_bob

*--SELECTS ALL THE STORES THAT HAS THE MOVIES HE APPEARS*

WHERE store\_id\_bob.address\_id = address.address\_id) AS city\_bob

*--SELECTS THE CITY ID IN WHICH THE STORE THAT HAS HIS FILMS ARE*

WHERE city\_bob.city\_id = city.city\_id

GROUP BY city.city\_id

ORDER BY city.city ASC *-- SELECTS THE NAME OF THE CITIES IN ALPHABETICAL ORDER*

1. Language in which most of the films have been filmed. The query must return the language.name attribute. If there is a tie between several languages, all languages in which the greatest number of films have been filmed should appear in the answer.

SELECT NAME

FROM language,

(SELECT language\_id,

order\_count

FROM (SELECT language\_id,

*Count*(\*) AS order\_count

FROM film

GROUP BY language\_id

ORDER BY *Count*(\*) DESC) AS aviable\_languages

*--SELECTS ALL THE AVAILABLE LANGUAGES*

WHERE order\_count IN (SELECT *Max*(order\_count)

FROM (SELECT language\_id,

*Count*(\*) AS order\_count

FROM film

GROUP BY language\_id

ORDER BY *Count*(\*) DESC) AS x)) AS

id\_movie *-- SELECTS THE TOP LANGUAGE IN WHICH MOVIES HAVE BEEN RECORD*

WHERE language.language\_id = id\_movie.language\_id *--SELECTS THE NAME OF THE TOP LANGUAGE MOVIES HAVE BEEN RECORD*

1. Language (of the films) in which a greater number of rentals has been done. The query must return the language.name attribute. If there is a tie between several languages, all languages in which a greater number of rentals has been done must appear in the answer.

SELECT z.NAME

FROM (SELECT x.NAME,

*Count*(\*) AS most\_rented\_movie

FROM (SELECT film\_rental.NAME

FROM rental,

(SELECT inventory\_id,

film\_inventory.NAME

FROM inventory,

(SELECT film\_id,

film\_name.NAME

FROM film,

(SELECT language\_id,

NAME

FROM language

GROUP BY language\_id) AS film\_name

WHERE film.language\_id = film\_name.language\_id)

AS

film\_inventory

*--SELECTS THE FILM ID AND THE NAME OF THE LANGAGE IT HAS BEEN RECORDED*

WHERE inventory.film\_id = film\_inventory.film\_id) AS

film\_rental

*--SELECTS THE ID OF THE INVENTORY THAT HAS THAT FILM*

WHERE film\_rental.inventory\_id = rental.inventory\_id) AS x

*--IF THE MOVIE HAS BEEN RENTED AND IT IS IN THE INVENTORY IT TAKES THE NAME OF IT*

GROUP BY x.NAME

ORDER BY most\_rented\_movie DESC) AS z

*--SELECTS ALL THE TIMES A FILM HAS BEEN RECORD IN A LANGUAGE*

WHERE most\_rented\_movie IN (SELECT *Max*(most\_rented\_movie)

FROM (SELECT x.NAME,

*Count*(\*) AS most\_rented\_movie

FROM (SELECT film\_rental.NAME

FROM rental,

(SELECT inventory\_id,

film\_inventory.NAME

FROM inventory,

(SELECT film\_id,

film\_name.NAME

FROM film,

(SELECT language\_id,

NAME

FROM language

GROUP BY language\_id) AS film\_name

WHERE film.language\_id =

film\_name.language\_id)

AS

film\_inventory

*--SELECTS THE FILM ID AND THE NAME OF THE LANGAGE IT HAS BEEN RECORDED*

WHERE inventory.film\_id = film\_inventory.film\_id) AS

film\_rental

*--SELECTS THE ID OF THE INVENTORY THAT HAS THAT FILM*

WHERE film\_rental.inventory\_id = rental.inventory\_id) AS x

*--IF THE MOVIE HAS BEEN RENTED AND IT IS IN THE INVENTORY IT TAKES THE NAME OF IT*

GROUP BY x.NAME

ORDER BY most\_rented\_movie DESC)AS y)*--Y=SELECTS ALL THE TIMES A FILM HAS BEEN RECORD IN A LANGUAGE*

*---PICKS THE TOP LANGUAGE A FILM HAS BEEN RECORDED*

1. Favorite category (category.name) of the customer who has rented more movies. By favorite category we refer to the category in which the client has made more rentals. If a client rents the same movie twice it should count as two rentals. The query must return the name (category.name) and the identifier (category id) of the category. If there is a tie between several clients, all clients who have rented more films should appear in the response.

CREATE OR replace VIEW customer\_and\_num\_rented\_movies

AS

SELECT customer\_id,

**Max**(count) AS num\_film

FROM (SELECT customer\_id,

category\_id,

**Count**(category\_id)

FROM (SELECT top\_customer.customer\_id,

film\_category.category\_id

FROM inventory,

rental,

film\_category,

(SELECT customer\_id

FROM (SELECT customer\_id,

**Count**(\*)

FROM rental

GROUP BY customer\_id

ORDER BY **Count** (\*) DESC) AS customer

WHERE count IN (SELECT **Max**(count)

FROM (SELECT customer\_id,

**Count**(\*)

FROM rental

GROUP BY customer\_id

ORDER BY **Count** (\*) DESC) AS

x)

)AS

top\_customer *--TOP CUSTOMERS*

WHERE rental.inventory\_id = inventory.inventory\_id

AND rental.customer\_id = top\_customer.customer\_id

AND inventory.film\_id = film\_category.film\_id) AS

custumer\_and\_category

*-- PICKS THE CATEGORY ID OF ALL TH EMOVIES THE TOP CUSTOMERS RENTED*

GROUP BY category\_id,

customer\_id

ORDER BY category\_id ASC) AS count\_category

*--COUNTS ALL THE TIMES A DIFFERENT CATEGORY HAS BEEN RENTED FROM A DIFFERENT CUSTOMER*

GROUP BY customer\_id; *-- IT GIVES YOU THE MOST RENTED CATEGORY OF THE TOP CUSTOMERS*

CREATE OR replace VIEW cost\_and\_category\_id\_count

AS

SELECT customer\_and\_num\_rented\_movies.customer\_id,

count\_category.category\_id,

count\_category.count

FROM customer\_and\_num\_rented\_movies,

(SELECT customer\_id,

category\_id,

**Count**(category\_id)

FROM (SELECT top\_customer.customer\_id,

film\_category.category\_id

FROM inventory,

rental,

film\_category,

(SELECT customer\_id

FROM (SELECT customer\_id,

**Count**(\*)

FROM rental

GROUP BY customer\_id

ORDER BY **Count** (\*) DESC) AS customer

WHERE count IN (SELECT **Max**(count)

FROM (SELECT customer\_id,

**Count**(\*)

FROM rental

GROUP BY customer\_id

ORDER BY **Count** (\*) DESC) AS

x)

)AS

top\_customer *--TOP CUSTOMERS*

WHERE rental.inventory\_id = inventory.inventory\_id

AND rental.customer\_id = top\_customer.customer\_id

AND inventory.film\_id = film\_category.film\_id) AS

custumer\_and\_category

*-- PICKS THE CATEGORY ID OF ALL TH EMOVIES THE TOP CUSTOMERS RENTED*

GROUP BY category\_id,

customer\_id

ORDER BY category\_id ASC) AS count\_category

*--COUNTS ALL THE TIMES A DIFFERENT CATEGORY HAS BEEN RENTED FROM A DIFFERENT CUSTOMER*

WHERE customer\_and\_num\_rented\_movies.num\_film = count\_category.count

AND customer\_and\_num\_rented\_movies.customer\_id =

count\_category.customer\_id *-- PICKS THE CUSTOMER ID AND THE CATEGORY ID AND SHOWS HOW MANY TIMES A CATEGORY HAS BEEN RENTED*

SELECT \* FROM customer\_and\_num\_rented\_movies

4. **DataBase Redesign**

In this exercise we were required to change the database so that if a staff member is transferred from one store to another, the information about the stores he had worked before are kept.

We added the following queries to the newdatabase.sql file:

﻿CREATE TABLE worked\_in (

staff\_id integer NOT NULL REFERENCES staff(staff\_id),

store\_id integer NOT NULL REFERENCES store(store\_id),

PRIMARY KEY (staff\_id, store\_id)

);

To create a new table that has as primary key the staff id and the store id. That are foreing keys at the same time.

INSERT INTO worked\_in VALUES

(1, 1);

(1, 2);

(2, 1);

We added those values at the table. The worker 1 has worked in the store 1 and 2. The number 2 has worked only in the first one.

1. We added the table **worked\_in** to store the required information. This table contains foreign keys connected to **staff** and **store**. 