Laura de Paz and Paula Samper
Data Structures, Group 125

REPORT OF ASSIGNMENT 1

DATA STRUCTURES

INTRODUCTION

In the first assignment of this subject we have put into practice the knowledge we have acquired in the theoretical lessons of this course about the creation and management of database using SQL.

In order to do so, we received a database called dvdrental that stores information about films, its cast and category, and where and who have rented them. In this assignment we had to understand and identify the primary as well as the foreign keys of the database and illustrate it using an ER diagram. Then we made some queries so that we can access certain information about the rentals.

First, we show here the DVD rental database tables, highlighting the primary and foreign keys, and how they are related to one another.

actor (actor_id, first_name, last_name, last_update)

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, sotore_id, first_name, last_name, email, address_id>address.address_id, activebool, create_date, last_update, active)

film (film_id, title, description, release_year, language_id->language.language_id, rental_durantion, rental_rate, length, replacement_cost, 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)

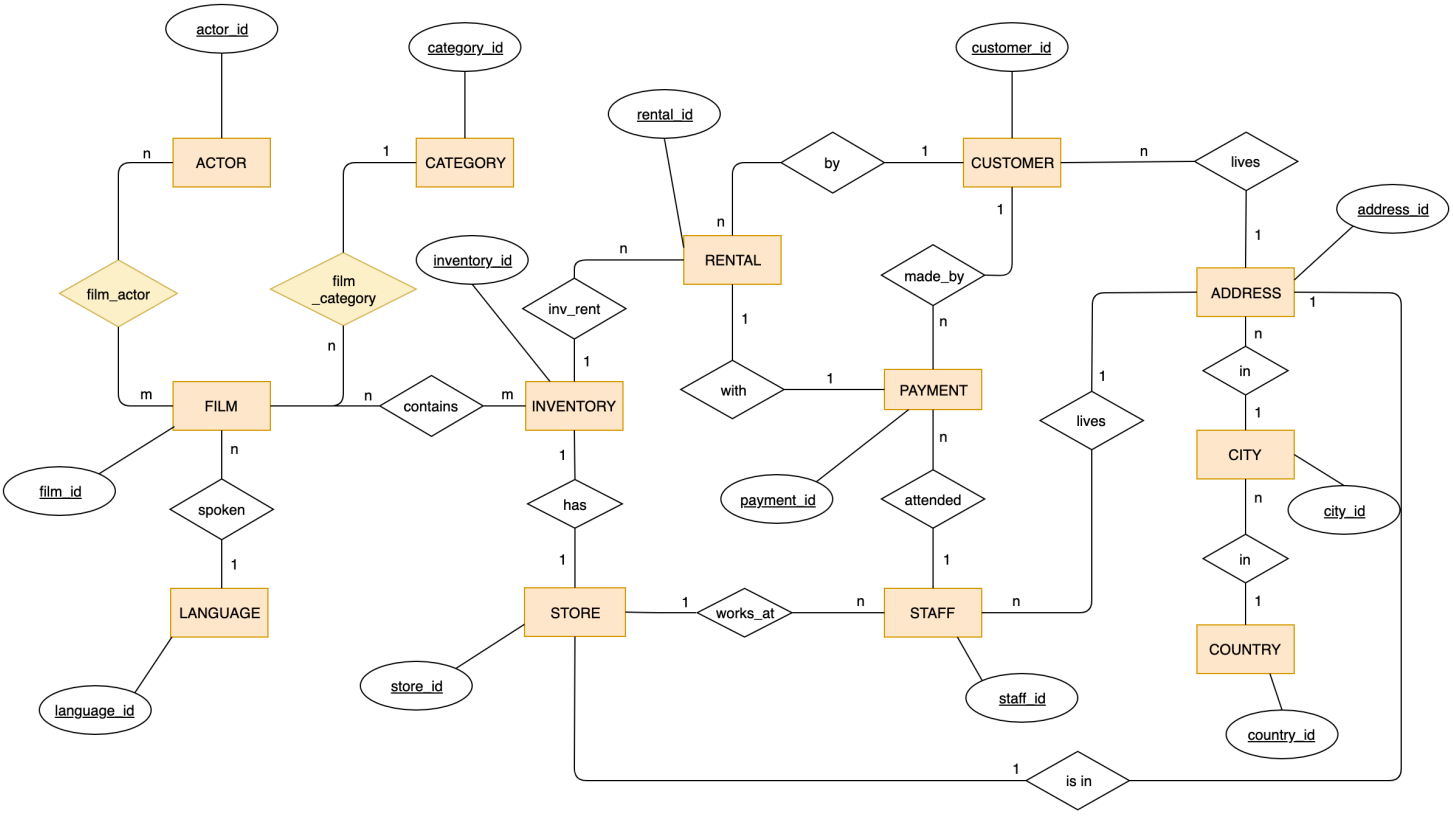
payment (payment_id, customer_id->customer.customer_id, staff_id->staff.staff_id, rental_id->rental.rental_id, amount, payment_date)

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, password, last_update, picture)

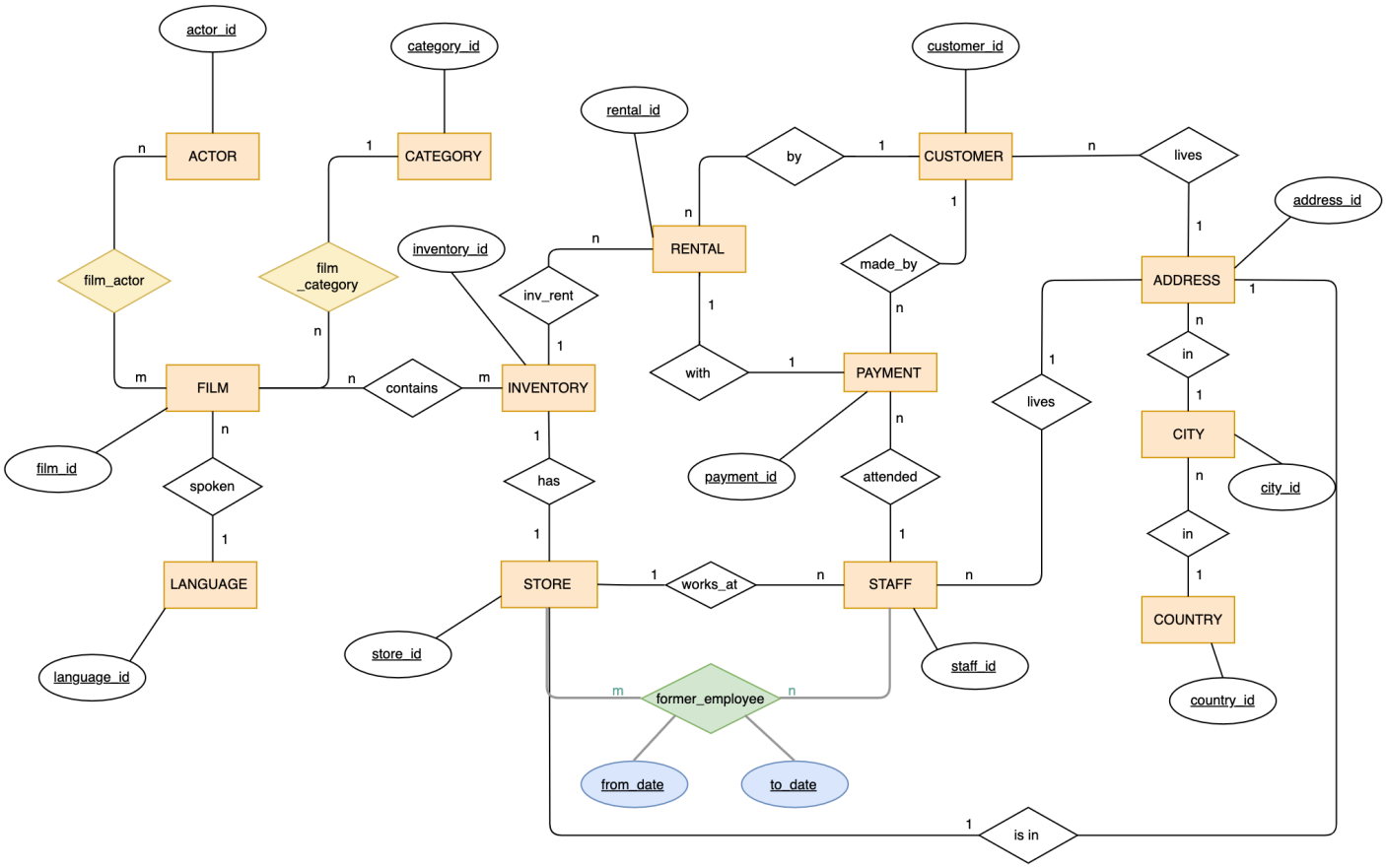
store (store_id, manager_staff_id->staf.staff_id, address_id->address.address_id, last_update)

Then we include the database relational schema:



Our next task is to redesign the database in order to keep record of the employees that used to work in each store, including the date of the first and the last day he/she worked there. Some changes have to be made in our database dvdrental:

- **New relational schema:**



As this diagram shows, we have added a new relational table to the database, in which we store information about where each employee has worked. This table has the following structure:

former_employee (staff_id->staff.staff_id, store_id->store.store_id, from_date, to_date last_update)

We added as well a new column in staff table to know when he started working at the store he is currently working at.

staff (staff_id, first_name, last_name, address_id->address.address_id, email, store_id, active, username, password, last_update, picture, **from_date**)

- **New option in the makefile so the command *make newdatabase* deletes all the tables in the database and recreate it with the new design.**

```
newdatabase:
	@echo Eliminando BBDD
	@$(DROPDB) $(DBNAME)
	rm -f *.log

	@echo Creando BBDD
	@$(CREATEDB)

	@echo restore data base
	@cat $(DBNAME).sql | $(PSQL)

	@echo adding information to newdatabase
	@cat newdatabase.sql | $(PSQL)
```

The instructions to create the new tables are stored in the file newdatabase.sql.

Lastly, we attach the queries we have been working on.

- **Query 1.** Number of movies rented each year.

```
SELECT Extract(year FROM rental_date) AS year,
       Count(*)
FROM   rental
GROUP  BY year
ORDER  BY year ASC;

-- We extract the year from the rental date and how many movies were rented that year,
-- and we order them in ascending order
```

- **Query 2.** Client who has rented more movies.

```
SELECT customer.customer_id,
       customer.first_name,
       customer.last_name,
       count(*) AS movies
FROM   rental,
       customer
WHERE  rental.customer_id = customer.customer_id --We relate the customer and the rental
GROUP  BY customer.customer_id -- We group and count the number of films by the customer_id
HAVING count(*) IN  (SELECT Count(*)
                     FROM   rental
                     GROUP  BY customer_id
                     ORDER  BY Count(*) DESC
                     LIMIT  1); -- This subquery counts the number of rentals per customer,
                                -- we order them in descending order and choose the first one
```

- **Query 3.** List the cities where movies, in which “Bob Fawcett” appears, have been rented.

```
 SELECT  DISTINCT city, --We use distinct so we get the city just once at the end
        city.city_id
 FROM   city,
        address,
        customer,
        rental,
        (SELECT rental_id
         FROM   actor,
                film_actor,
                inventory,
                rental
         WHERE  actor.first_name = 'Bob'
                AND actor.last_name = 'Fawcett'
                AND actor.actor_id = film_actor.actor_id
                AND film_actor.film_id = inventory.film_id
                AND inventory.inventory_id = rental.inventory_id) AS bf
                -- We create a subquery to relate the actor to the rentals of the films in
                -- which this actor has worked. We call it bf.

 WHERE  city.city_id = address.city_id
        AND address.address_id = customer.address_id
        AND customer.customer_id = rental.customer_id
        AND rental.rental_id = bf.rental_id -- We find the intersection between the rentals of a city and bs
 ORDER  BY city;
 --We order the resulting list by the city from A to Z by default
```

- **Query 4.** Language in which most of the films have been filmed.

```
SELECT language.name
FROM   language,
       (SELECT language_id,
               Count(*) AS num
        FROM   film
        GROUP  BY language_id
        ORDER  BY num DESC
        LIMIT  1) AS top_language
         -- This subquery counts the number of times a language is related to a film
         -- (the language in which the film has been filmed),and chooses the most popular one
WHERE  top_language.language_id = language.language_id;
-- We relate the most popular language with the language.id
```

- **Query 5.** Language (of the films) in which a greater number of rentals has been done.

```
SELECT language.name
FROM   language,
       (SELECT film.language_id,
               Count(*)
        FROM   rental,
               inventory,
               film
        WHERE  rental.inventory_id = inventory.inventory_id
               AND inventory.film_id = film.film_id
        GROUP  BY film.language_id
        ORDER  BY Count(*)  DESC
        LIMIT  1) AS top_lang
        -- In this subquery we relate the rentals with the films, to find out the language in
        -- which the rental has been done, counts them, and chooses the most popular one.
        -- We call it top_lang
WHERE  top_lang.language_id = language.language_id;
-- We relate top_lang with language.id
```

- **Query 6.** Favorite category of the customer who has rented more movies.

```
CREATE view top_rent_cat
AS
  (SELECT rental.customer_id,
          category.name,
          Count(*) AS total
   FROM   rental,
          inventory,
          film,
          film_category,
          category,
          (SELECT rental.customer_id AS top,
                  Count(*)           AS num
           FROM   rental
           GROUP  BY rental.customer_id
           ORDER  BY num DESC
           LIMIT  1) AS top_rent
           -- We find the customer that has made more rentals by counting the times that the
           -- same rental.customer_id appears and choosing the one that appears the most (top_rent)
   WHERE  rental.inventory_id = inventory.inventory_id
          AND inventory.film_id = film.film_id
          AND film.film_id = film_category.film_id
          AND film_category.category_id = category.category_id
          AND rental.customer_id = top_rent.top
   GROUP  BY rental.customer_id,
             category.name);
             --We relate the rental to the category and the rental.customer to top_rent

SELECT name
FROM   top_rent_cat
ORDER  BY total DESC
LIMIT  1;
--We select the category with the highest amount of rentals(of the top_rent)

DROP VIEW top_rent_cat;
```