

DATA STRUCTURES ASSIGNMENT 2:

DATABASES AND PROGRAMMING LANGUAGES

Samai García González and S. Xiao Fernández Marín

Group 1292

Pair 1

INTRODUCTION

The objective of this assignment is to create our own programs in order to interact with our database from the previous task. For this, we will create several programs in C language that will read our queries in SQL. We will do everything using the ODBC library.

Program 1: DVDREQ

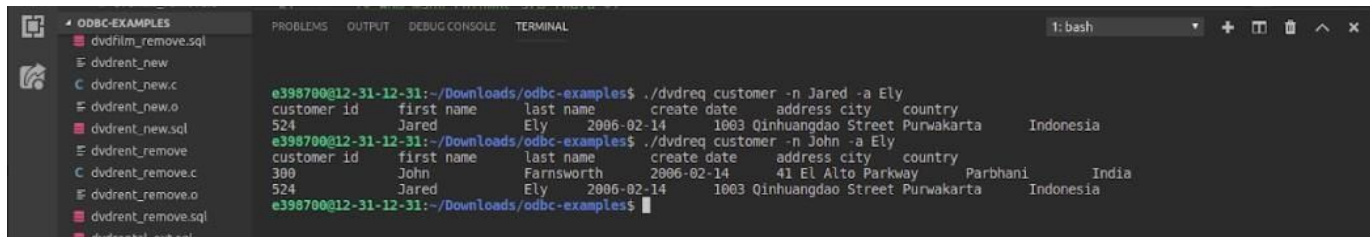
For this program, we wanted it to allow us to perform the following queries on the database:

dvdreq customer -n <First Name> -a <Last Name>

This query first checks the arguments and then prints a list of customer with the same first name, or last name or both, as the ones entered by the user.

```
SELECT customer_id,
       first_name,
       last_name,
       create_date,
       address,
       city,
       country
FROM   customer,
       address,
       city,
       country
WHERE  city.country_id = country.country_id
AND    city.city_id = address.city_id
AND    address.address_id = customer.address_id
AND    ( first_name = '?'
        OR last_name = '?' )
```

This is the result when executing it:



```
e398700@12-31-12-31:~/Downloads/odbc-examples$ ./dvdreq customer -n Jared -a Ely
customer_id  first_name  last_name  create_date  address      city      country
524         Jared      Ely        2006-02-14   1003 Qinhuangdao Street Purwakarta  Indonesia
e398700@12-31-12-31:~/Downloads/odbc-examples$ ./dvdreq customer -n John -a Ely
customer_id  first_name  last_name  create_date  address      city      country
390         John      Farnsworth  2006-02-14   41 El Alto Parkway Parbhani    India
524         Jared      Ely        2006-02-14   1003 Qinhuangdao Street Purwakarta  Indonesia
e398700@12-31-12-31:~/Downloads/odbc-examples$
```

dvdreq film <title>

This query, first checks that the title of the film entered as an argument exists:

```
SELECT title
FROM film
WHERE title = '%s';
```

Then, if it exists, it prints every movie in the database whose title matches, fully or partially, the entered title:

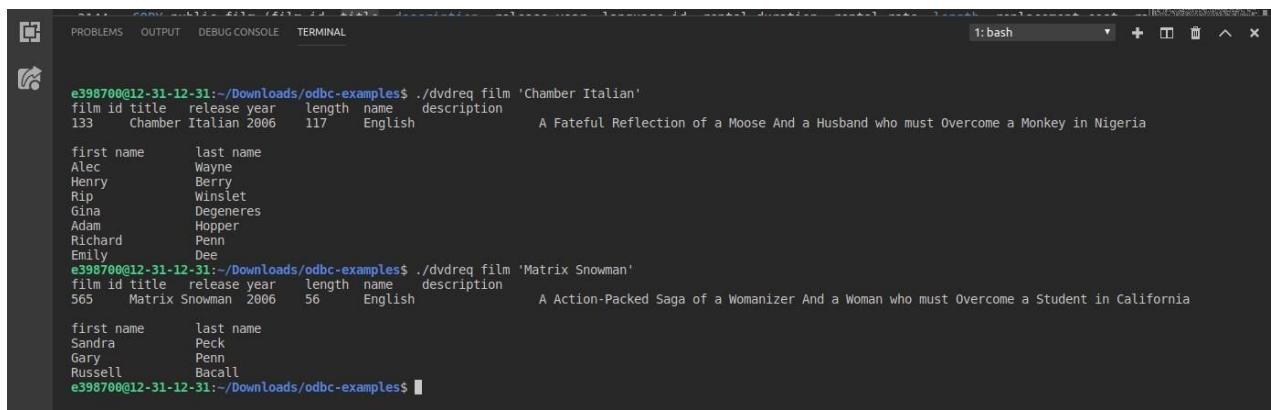
This prints all the information about the film:

```
SELECT film.film_id,
       title,
       release_year,
       length,
       NAME,
       description
FROM film,
     language,
     film_actor,
     actor
WHERE language.language_id = film.language_id
AND actor.actor_id = film_actor.actor_id
AND film.film_id = film_actor.film_id
AND film_actor.film_id = film.film_id
AND title = '?'
```

This prints the first and last name of all the actors of the film.

```
SELECT first_name,
       last_name
FROM film,
     LANGUAGE,
     film_actor,
     actor
WHERE LANGUAGE.language_id = film.language_id
AND actor.actor_id = film_actor.actor_id
AND film.film_id = film_actor.film_id
AND film_actor.film_id = film.film_id
AND title = '%s';
```

This is the result while executing it:



```
e398700@12-31-12-31:~/Downloads/odbc-examples$ ./dvdreq film 'Chamber Italian'
film id title  release year  length name  description
133  Chamber  Italian  2006  117  English  A Fateful Reflection of a Moose And a Husband who must Overcome a Monkey in Nigeria

first name  last name
Alec        Wayne
Henry       Berry
Rip         Winslet
Gina        Degeneres
Adam        Hopper
Richard     Penn
Emily       Dee

e398700@12-31-12-31:~/Downloads/odbc-examples$ ./dvdreq film 'Matrix Snowman'
film id title  release year  length name  description
565  Matrix  Snowman  2006  56  English  A Action-Packed Saga of a Womanizer And a Woman who must Overcome a Student in California

first name  last name
Sandra      Peck
Gary        Penn
Russell     Bacall
e398700@12-31-12-31:~/Downloads/odbc-examples$
```

dvdreq rent <customer_id> <init date> <end date>

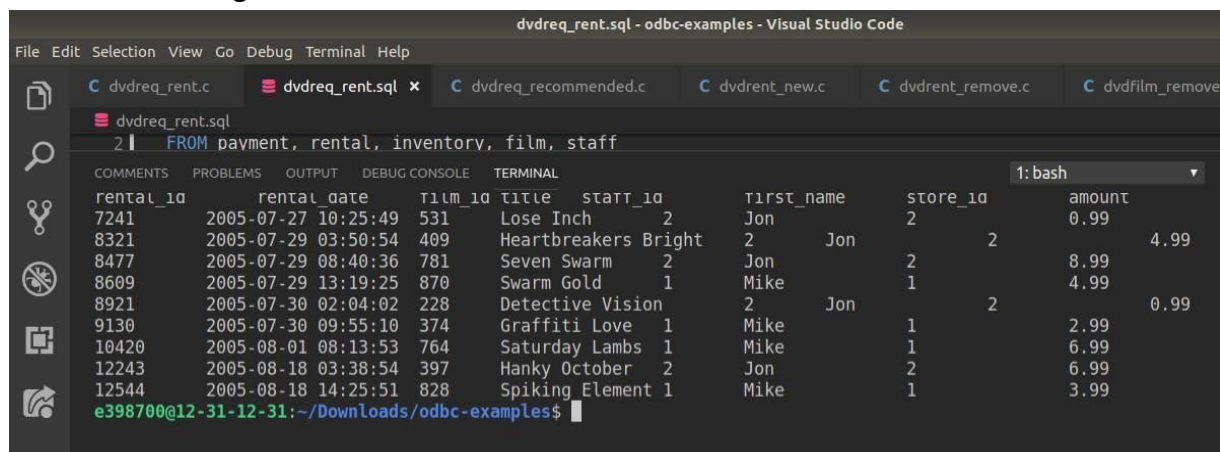
This query first checks that the customer_id received as an argument exists:

```
SELECT customer_id
FROM   rental
WHERE  customer_id = %d;
```

Then, prints the rentals that the customer entered has carried out between the init and end dates entered. It prints the rentals sorted by rental date.

```
SELECT rental.rental_id,
       rental_date,
       film.film_id,
       title,
       staff.staff_id,
       first_name,
       staff.store_id,
       amount
FROM   rental,
       inventory,
       film,
       staff,
       payment
WHERE  rental.inventory_id = inventory.inventory_id
AND    payment.staff_id = staff.staff_id
AND    payment.rental_id = rental.rental_id
AND    inventory.film_id = film.film_id
AND    rental.customer_id = %i
AND    rental_date < '%s'
AND    rental_date > '%s';
```

This is the running result:



rental_id	rental_date	film_id	title	staff_id	first_name	store_id	amount
7241	2005-07-27 10:25:49	531	Lose Inch	2	Jon	2	0.99
8321	2005-07-29 03:50:54	409	Heartbreakers Bright	2	Jon	2	4.99
8477	2005-07-29 08:40:36	781	Seven Swarm	2	Jon	2	8.99
8609	2005-07-29 13:19:25	870	Swarm Gold	1	Mike	1	4.99
8921	2005-07-30 02:04:02	228	Detective Vision	2	Jon	2	0.99
9130	2005-07-30 09:55:10	374	Graffiti Love	1	Mike	1	2.99
10420	2005-08-01 08:13:53	764	Saturday Lambs	1	Mike	1	6.99
12243	2005-08-18 03:38:54	397	Hanky October	2	Jon	2	6.99
12544	2005-08-18 14:25:51	828	Spiking Element	1	Mike	1	3.99

dvdreq recommend <customer Id>

First of all, we again check that the customer id exists:

```
SELECT customer_id
FROM rental
WHERE customer_id = %d;
```

Then, we create a list with all the film categories that the customer with the entered customer_id has rented:

```
CREATE OR replace VIEW list_rented
AS
    SELECT category_id,
           Count(category_id)
    FROM film_category,
         (SELECT film_id
          FROM rental,
              inventory
          WHERE rental.customer_id = %i
              AND rental.inventory_id = inventory.inventory_id) AS
        customer_rented
    WHERE customer_rented.film_id = film_category.film_id
    GROUP BY category_id
    ORDER BY count DESC;
```

After that, it creates a list with the three recommended films that belong to the most rented category of the customer, but that have not been rented before by the customer.

```
CREATE OR replace VIEW most_rented_films AS SELECT film_id
FROM (
    SELECT film_category.film_id,
           Count(rental.inventory_id)
    FROM film_category,
         inventory,
         rental,
         (
             SELECT list_rented.category_id
             FROM list_rented,
                  (
                      SELECT count
                      FROM list_rented LIMIT 1) AS max
             WHERE list_rented.count = max.count) AS
        most_rented_category --This query selects the most rented category from the customer
```

```
    WHERE film_category.category_id = most_rented_category.category_id
    AND film_category.film_id = inventory.film_id
    AND inventory.inventory_id = rental.rental_id
    GROUP BY film_category.film_id
    ORDER BY count DESC ) AS x
EXCEPT
```

```

        (
            SELECT film_i)
from    rental,
        inventory
WHERE   rental.customer_id = %i
AND     rental.inventory_id = inventory.inventory_id) limit 3;

```

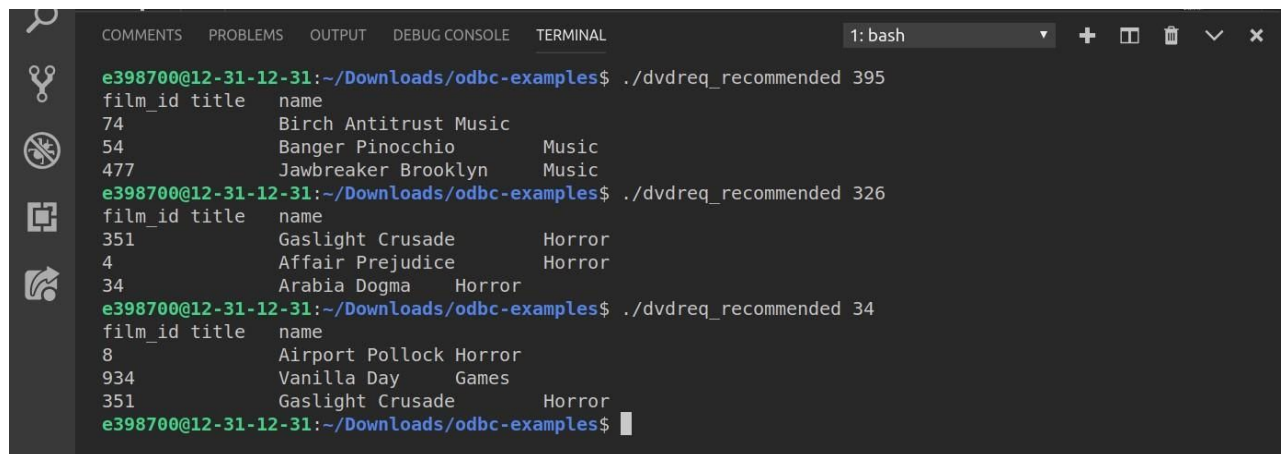
Finally, it prints the film_id, title and category of the three recommended films.

```

SELECT most_rented_films.film_id,
       film.title,
       category.name
FROM   most_rented_films,
       film,
       film_category,
       category
WHERE  most_rented_films.film_id = film.film_id
AND    film_category.film_id = most_rented_films.film_id
AND    film_category.category_id = category.category_id

```

This is the result of running the program:



```

e398700@12-31-12-31:~/Downloads/odbc-examples$ ./dvdreq_recommended 395
film_id title      name
74      Birch Antitrust Music
54      Banger Pinocchio Music
477     Jawbreaker Brooklyn Music
e398700@12-31-12-31:~/Downloads/odbc-examples$ ./dvdreq_recommended 326
film_id title      name
351     Gaslight Crusade Horror
4       Affair Prejudice Horror
34      Arabia Dogma Horror
e398700@12-31-12-31:~/Downloads/odbc-examples$ ./dvdreq_recommended 34
film_id title      name
8       Airport Pollock Horror
934     Vanilla Day Games
351     Gaslight Crusade Horror
e398700@12-31-12-31:~/Downloads/odbc-examples$

```

Program 2: DVDRENT

The objective of this program is to manage the rentals in the database. The following queries were performed:

dvdrent new <customer id> <film id> <staff id> <store id> <amount>

First checks that there is a film with that film_id, and that the film exists in the inventory so it is available for renting.

```
SELECT film.film_id,  
       inventory_id  
FROM   film,  
       inventory  
WHERE  film.film_id = %i  
       OR inventory.film_id = %i
```

Then, it checks that the customer_id, staff_id and store_id exist:

```
SELECT customer_id  
FROM   customer  
WHERE  customer_id = ?;
```

```
SELECT staff_id  
FROM   staff  
WHERE  staff_id = ?;
```

```
SELECT store_id  
FROM   store  
WHERE  store_id = ?;
```

After that, it inserts the values of the arguments in the rental and payment table:

```
INSERT INTO rental  
VALUES      (DEFAULT,  
            Now(),  
            %i,  
            %i,  
            NULL,  
            %i,  
            Now())
```

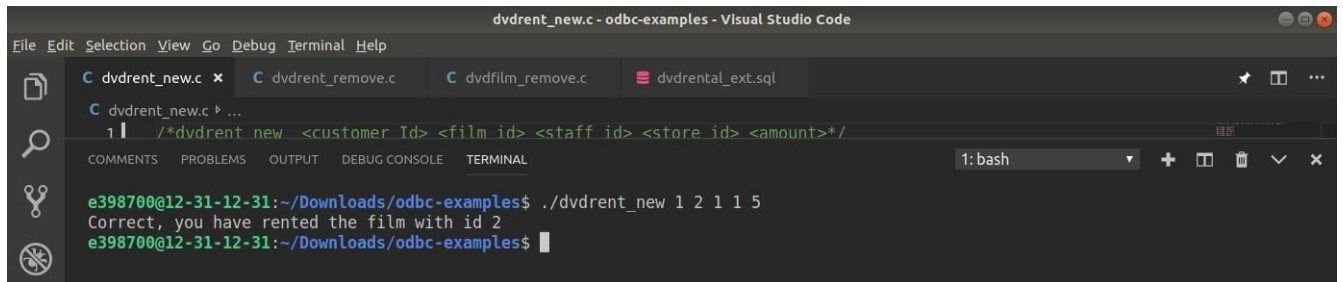
We pick the rental_id from the new rental we created:

```
SELECT rental_id  
FROM   rental  
WHERE  inventory_id = %s
```

We add it to the payment table:

```
INSERT INTO payment
VALUES      (DEFAULT,
            %i,
            %i,
            %s,
            %i,
            Now ( ) )
```

This is the result:

A screenshot of the Visual Studio Code editor interface. The top bar shows the title 'dvdrent_new.c - odbc-examples - Visual Studio Code'. The left sidebar contains icons for Explorer, Search, and Run and Debug. The main editor area has a file explorer on the left showing 'dvdrent_new.c', 'dvdrent_remove.c', 'dvdrent_remove.c', and 'dvdrent_ext.sql'. The 'dvdrent_new.c' file is open, showing a comment: '/*dvdrent_new <customer Id> <film id> <staff id> <store id> <amount>*/'. Below the editor, the 'TERMINAL' tab is active, showing a bash shell. The terminal output shows the command './dvdrent_new 1 2 1 1 5' being executed, followed by the message 'Correct, you have rented the film with id 2' and the prompt 'e398700@12-31-12-31:~/Downloads/odbc-examples\$'.

dvdrent remove <rent Id>

First, we check that the rental_id entered exists in rental or payment:

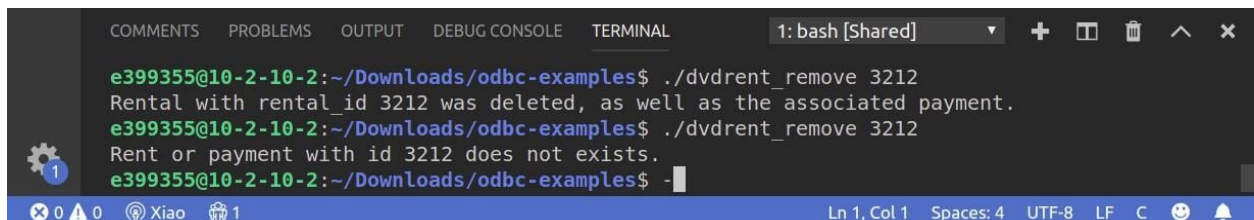
```
SELECT payment.rental_id FROM rental,
       payment
WHERE  rental.rental_id = payment.rental_id
      AND rental.rental_id = %i
```

Then, it removes the rental associated to that rental_id from the database and its associated payment.

```
DELETE FROM payment
WHERE  payment.rental_id = %i

DELETE FROM rental
WHERE  rental_id = ?
```

This is the result obtained when running it:

A screenshot of a terminal window. The terminal shows the command './dvdrent_remove 3212' being executed. The output is 'Rental with rental_id 3212 was deleted, as well as the associated payment.' followed by another command './dvdrent_remove 3212' which results in the message 'Rent or payment with id 3212 does not exists.' The terminal prompt is 'e399355@10-2-10-2:~/Downloads/odbc-examples\$'.

Program 3: DVDFILM

This program basically has one query, and it is in charge of removing a film from the database.

dvdfilm remove <film id>

First it checks that there is a film in the database with that film_id:

```
SELECT film_id
FROM film
WHERE film_id = %i
```

Then, it removes everything related to the film:

- From payment:

```
DELETE FROM payment
WHERE payment_id IN (SELECT payment_id
                     FROM payment,
                     rental,
                     inventory
                     WHERE inventory.inventory_id = rental.inventory_id
                     AND payment.rental_id = rental.rental_id
                     AND film_id = %i);
```

- From rental:

```
DELETE FROM rental
WHERE rental_id IN (SELECT rental_id
                   FROM rental,
                   inventory
                   WHERE inventory.inventory_id = rental.inventory_id
                   AND film_id = %i);
```

- From inventory, film category, film_actor, film:

```
DELETE FROM inventory
WHERE film_id = %i

DELETE FROM film_category
WHERE film_id = %i

DELETE FROM film_actor
WHERE film_id = %i

DELETE FROM film
WHERE film_id = %i
```

This is the result when running it:



```
1: bash [Shared]
e399355@10-2-10-2:~/Downloads/odbc-examples$ ./dvdfilm_remove 23
Film with rental id 23 was deleted.
e399355@10-2-10-2:~/Downloads/odbc-examples$ ./dvdfilm_remove 23
Film with id 23 does not exists.
e399355@10-2-10-2:~/Downloads/odbc-examples$
```

CONCLUSION

In this practice we have learned how to implement queries in sql to c programming. The results obtained were satisfying.