

Universidad Politécnica de Aguascalientes

Ingeniería en Sistemas Computacionales

Administración de Base de Datos

Teacher: Juan Carlos Herrera Hernández

ISC06A

Rafael Martinez Verdin UP200190

Cristian Alessandro Verdin Mata UP200220

Paola Castañeda Serrano UP200264

Hector Enrique Aguayo Garcia UP200428

Obet Leonardo Fuentes Ontiveros UP200677

Eduardo Pulido Guzmán UP200516

Delivery Date : 25/10/2022

INDEX

Objective

A database allows to store a great amount of information in an organized way for its future consultation, searches, new data entry, etc. All this can be done in a quick and simple way from a computer. Then at the moment of making this database of a movie rental store we have as objective to help the store to have a better control of sales and results at the moment of making some consultation of an article.

A properly designed database allows you to gain access to up-to-date and accurate information. Since it is essential to have a correct design to achieve your goals of working with a database, it makes sense to invest the time necessary to learn about the principles of good design.

Introduction

In this database called "Online Media Rentals" there is a wide variety of movies where we can find them by category, description, rating, and release date. We can find certain customers with their respective data for each client. However we have a table that divides the movies by the format of each movie, we have the table of actors in order to identify which actor acts in each movie and the respective data of each actor, finally another table that has the rental history where we can find the rental date, the customer who rented the movie and when you have to return the movie.

In this document we are going to present a structure of certain steps that we have been required to perform this project, we will include some organization to be clear in each step that is required.

The objective of this database is to have a set of administration of the movies that are rented in a movie rental store, so there is a control of which movie each customer rents and when he has to return the movie, and thus providing the best service to each customer.

To achieve the objective of this project, we met as a team and structured an organization dividing certain parts of the project in order to obtain the best possible result.

STEPS ORACLE

Flix User Creation:

Before anything else for the database, we first need a user who will be in charge of manipulating the desired database. For this, from the SYS administrator, a user is created granting him certain permissions so that he can act in a certain way with the database.

- **ALL PRIVILEGES:** This would grant a MySQL user full access to a designated database (or if no database is selected, global access to the entire system).
- **CREATE:** Allows you to create new tables or databases.
- **DROP:** Allows you to delete tables or databases.
- **DELETE:** Allows you to delete rows from tables.
- **INSERT:** Allows you to insert rows in tables.
- **SELECT:** Allows you to use the SELECT command to read the databases.
- **UPDATE:** Allows you to update the table rows.
- **GRANT OPTION:** Allows you to grant or remove privileges from other users.

Create tables:

- In order to create the tables, a hierarchy had to be followed. it was necessary to create the primary tables since the secondary tables depend on them.
- One of the most important parts is creating the constraints.
- Once the primary tables have been created, we proceed to create the secondary tables.

- When creating the secondary tables it was necessary to make the links or foreign keys.

Creating Views:

It was necessary to create a view to show the title and the media_id of the movies that have not yet been returned.

Sequence:

Something important was the sequences to have an order in the assignment of our primary keys. Often this is the primary key field that we would like to be created automatically every time a new record is inserted. Sequence numbers are generated independently of tables, so the same sequence can be used for one or for multiple tables. The sequence was also used for the exercises indicated in the image found in the MySQL sequence section.

Add to data to the tables:

To add data to the tables, the INSERT INTO statement was used, making sure to use the sequences.
We use “Insert into” then the name of the table, we detail the names of the fields in parentheses and separated by commas and after the “values” clause we put the values for each field, also in parentheses and separated by commas.

For the basic command of data insertion we did not have any difficulty, only it was necessary to do the research for the format of dates and sequences of numbers in Oracle.

It is important to remember that whenever we insert data it must be in tables that do not have foreign key restrictions, otherwise if the table where the foreign key comes from does not have records or the data does not match the parent table, it will give us an error because it does

not exist or it cannot find it, therefore it is important to take into account to insert data in tables that do not have foreign key restrictions.

It is important to take into account that for our records to follow the sequences we specify we have to create the sequence and use **ID_NAME.NEXTVAL** so that we do not insert the id directly.

In Oracle, TO_DATE function converts a string value into DATE data type value using the specified format.

As shown in this example:

- **TO_DATE**(string, format).
- **TO_DATE**('6/10/2022','DD/MM/YYYY')

And also depending on how we have the format in our database there is a default format that is specified by **NLS_DATE_FORMAT** that can recognize many formats.

Index:

It is important to make an index to speed up searches, so we choose to create it in the customers table.

Create a synonym(ORACLE):

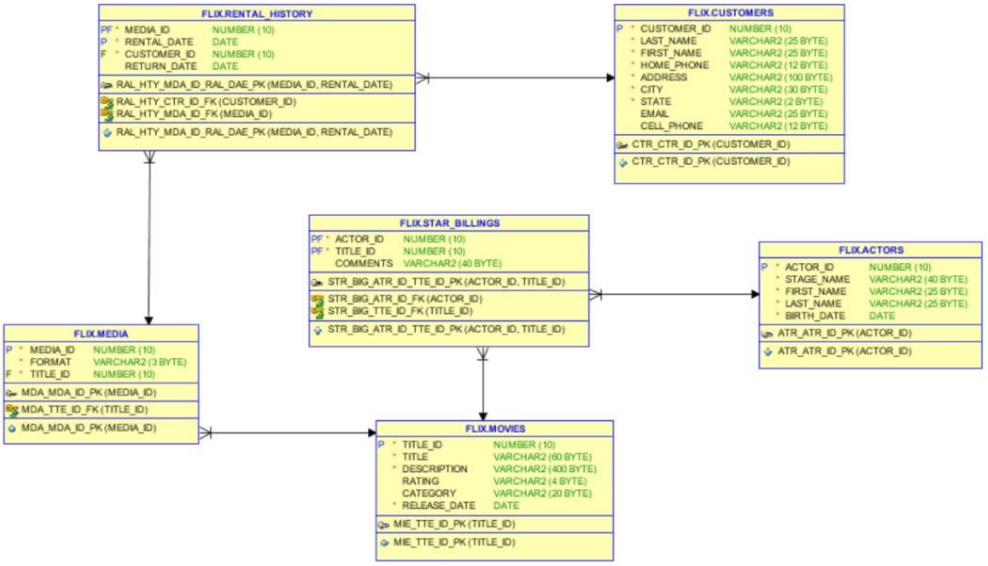
The creation of the synonym for the "TITLE_UNAVAIL" table is simply to represent it. It is used to be able to refer to the table without having to prepend its schema. A public synonym can be seen by all users, but a private synonym can only be seen by the user who created it.

https://www.ibm.com/docs/es/rtw/9.1.1?topic=overview-synonyms

Diagrams

We can see that as a final result we arrive at the following entity relationship diagram, in which the efficiency of the relationships between tables can be verified.

Diagram(Oracle)



QUERYS

ORACLE

Problem 1:

We need to know who is a frequent customer and who is not.
It is known that 4 movie rentals is considered as a frequent customer.
keep in mind that frequent client "CF" and infrequent client is "NF".

CODE:

```
select  customer_id, last_name,
case when (customer_id in (select customer_id from rental_history
                           GROUP by customer_id
                           HAVING count(customer_id)>3) )then 'CF'
else 'NF' end tipo
from customers;
```

Result:

	CUSTOMER_ID	LAST_NAME	TIPO
1	101	Raynard	CF
2	102	Records	NF
3	103	Hesbrook	NF
4	104	Fulloway	NF
5	105	Crips	NF
6	106	Skeermor	NF
7	107	Hischke	NF
8	108	Clackson	NF
9	109	Dionisetto	NF
10	110	Wagon	NF
11	111	Wisson	NF
12	112	Sackey	CF
13	113	Pimblett	NF
14	114	Towll	NF
15	115	Hickin	NF
16	116	Melhuish	NF
17	117	Keedwell	NF
18	118	Cordelle	NF
19	119	Schoffler	NF

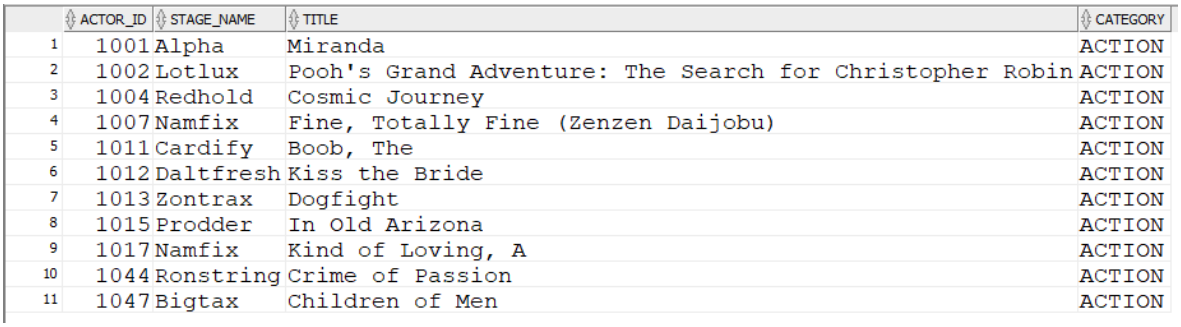
Problem 2:

Shows the artistic name of the actors who participated in action category movies (show movie title).

CODE:

```
SELECT a.actor_id, a.stage_name, m.title, m.category FROM movies m
INNER JOIN star_billings s ON m.title_id = s.title_id
INNER JOIN actors a ON s.actor_id = a.actor_id
WHERE category = 'ACTION';
```

Result:



	ACTOR_ID	STAGE_NAME	TITLE	CATEGORY
1	1001	Alpha	Miranda	ACTION
2	1002	Lotlux	Pooh's Grand Adventure: The Search for Christopher Robin	ACTION
3	1004	Redhold	Cosmic Journey	ACTION
4	1007	Namfix	Fine, Totally Fine (Zenzen Daijobu)	ACTION
5	1011	Cardify	Boob, The	ACTION
6	1012	Daltfresh	Kiss the Bride	ACTION
7	1013	Zontrax	Dogfight	ACTION
8	1015	Prodder	In Old Arizona	ACTION
9	1017	Namfix	Kind of Loving, A	ACTION
10	1044	Ronstring	Crime of Passion	ACTION
11	1047	Bigtax	Children of Men	ACTION

Problem 3:

Show the categories that were rented per month

CODE:

```
select mo.category , to_char(r.rental_date, 'MONTH') as mes
from movies mo
inner join media m on mo.title_id = m.title_id
inner join rental_history r on m.media_id = r.media_id;
```

Result:

	CATEGORY	MES
1	ACTION	JANUARY
2	ACTION	MARCH
3	ACTION	MARCH
4	CHILD	JANUARY
5	ACTION	JANUARY
6	ACTION	JULY
7	ACTION	MAY
8	DRAMA	NOVEMBER
9	CHILD	APRIL
10	CHILD	AUGUST
11	DRAMA	JANUARY
12	COMEDY	NOVEMBER
13	ACTION	JANUARY
14	ACTION	JUNE
15	CHILD	JUNE
16	ACTION	NOVEMBER
17	ACTION	JUNE
18	DOCUMENTARY	MAY
19	DOCUMENTARY	MAY

STEPS MySQL

Flix User Creation:

MySQL is an open source database management software that helps users store, organize and retrieve data. It has several options for granting nuanced permissions to specific users on tables and databases.

Here is a brief list of other possible common permissions that users can obtain.

Create Tables:

The tables are the most important elements within a database since they are used to store the data of our application.
A table is mainly made up of rows and columns. In it, the columns are called fields, while the rows are called records. We can create a table using the command CREATE TABLE. The command CREATE TABLE requires you to specify the name of the table and its header, which is a comma-separated list of field definitions.

Also, note that each “field definition” is made up of the field name, data type, and constraints.

Creating Views:

It was necessary to create a view to show the title and the media_id of the movies that have not yet been returned.

Sequence:

In MySQL we create a sequence to obtain a unique identifier for each row and automatically increment it. The simplest MySQL method is to use a sequence to define the use of the MySQL AUTO_INCREMENT column.

In this project we used the sequences indicated below:

- Use a sequence to generate PKs for CUSTOMER_ID in CUSTOMERS table
  - Begin at 101 and increment by 1
- Use a sequence to generate PKs for TITLE_ID in MOVIES table
  - Begin at 1 and increment by 1
- Use a sequence to generate PKs for MEDIA_ID in MEDIA table
  - Begin at 92 and increment by 1

- Use a sequence to generate PKs for ACTOR_ID in ACTOR table
  - Begin at 1001 and increment by 1
- Run queries from the data dictionaries for the above sequences.

Add to data to the tables:

For MySQL the syntax of the data insertion is the same, the only thing that changes in this case compared to Oracle is the operation of the number sequences and the date format works the same, only the sentence changes. As for the data insertion it is the same as Oracle when we have the restriction of a foreign key.

Talking about the operation of a sequence does not provide any built-in function to create a sequence for rows or columns of a table. But we can generate it through a MySQL query , at the time of performing the data insertion we just delete the field and since when we create the table and add the **AUTO_INCREMENT** to the INT type field it will follow a numeric sequence. Normally used in primary Keys.

In **MySQL**, STR_TO_DATE converts the string to a date value based on the format string. The function can return a, or a value based on the input and format strings.

As shown in this example:

- **STR_TO_DATE**(string, format);
- **STR_TO_DATE**('21,5,2013','%d,%m,%Y');

1. It tries to find a match for the format specifier, which is a day.

2. Because of the comma literal character (,), the function continues to check the second format specifier, which is a month.
3. After matching the second comma (,), the function continues to find a match for the third format specifier, which is a four-digit year.

Index:

It is important to make an index to speed up searches, so we choose to create it in the customers table.

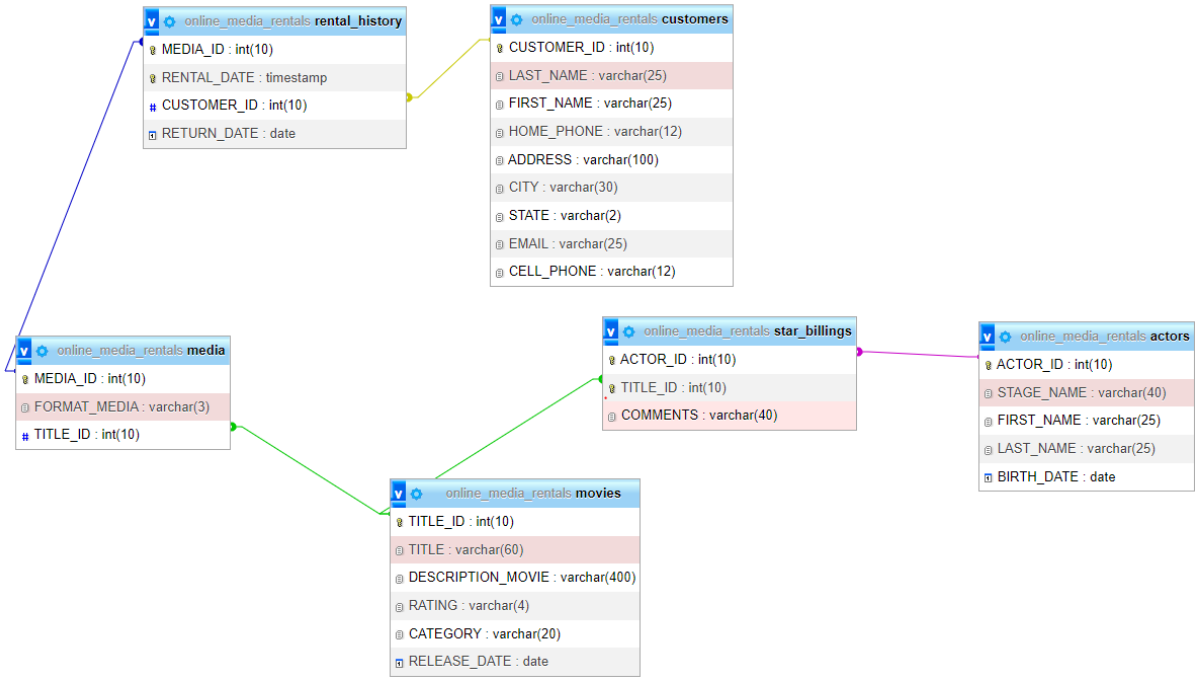
Create a synonym:

It is not applicable since MySQL does not support synonyms, since there is no instruction related to synonyms and to argue this in the following link we can corroborate it.
https://www.ibm.com/docs/es/rtw/9.1.1?topic=overview-synonyms

Diagrams

We can see that as a final result we arrive at the following entity relationship diagram, in which the efficiency of the relationships between tables can be verified.

Diagram (MySQL)



Conclusion

In conclusion, creating a database from scratch was a somewhat tedious but not difficult job. Working as a team is helpful because that way we can discuss points for a better development of the database. Something important that we can add is that databases entail three different roles which are the designers who manage the data, the developer who implements interfaces and transactions and the end users who can manipulate and edit the data.