Universidad Politécnica de Aguascalientes

Juan Carlos Herrera Hernandez

Administración de Base de Datos

Integrantes:

Flores Andrade Ixchel Alejandra	UP200417
Ruiz Ortiz Cristopher Kaleb	UP200674
Landeros Angel Emmanuel	UP200518
Valadez Lariz Jaime Jesabel	UP200205
Sanchez Olvera Diego	UP210010
Espinoza Magaña Oliver	UP201005
Hernández Macías César Alejandro	UP200775

Documentation

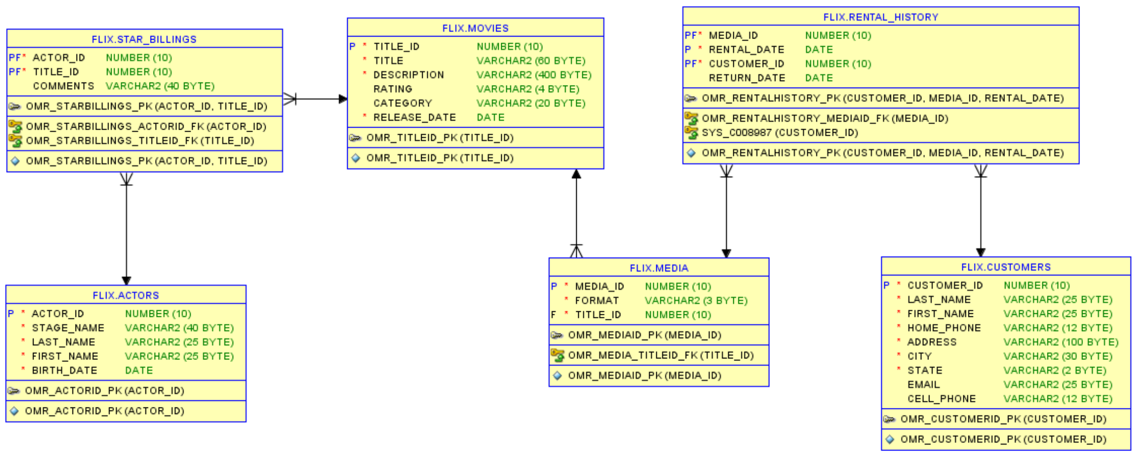
Online Media Rental

The Online Media Rental database has the purpose of storing and managing the data of a movie rental store, dividing them into several tables that are related and with which you can have a control of the necessary data of customers, movies, rentals, among others.

This database has the necessary restrictions and connections to avoid erroneous data and to have a good control of the rentals to facilitate the management of all the information of the store.

Oracle

The final design of the database in Oracle is presented by means of six tables, which were named Actors, Movies, Media, Rental History, Star Billings and Customers. The diagram is shown in the following figure:


FLIX.STAR_BILLINGS
PF* ACTOR_ID NUMBER (10)
PF* TITLE_ID NUMBER (10)
COMMENTS VARCHAR2 (40 BYTE)
OMR_STARBILLINGS_PK (ACTOR_ID, TITLE_ID)
OMR_STARBILLINGS_ACTORID_FK (ACTOR_ID)
OMR_STARBILLINGS_TITLEID_FK (TITLE_ID)
OMR_STARBILLINGS_PK (ACTOR_ID, TITLE_ID)
FLIX.MOVIES
P * TITLE_ID NUMBER (10)
* TITLE VARCHAR2 (60 BYTE)
* DESCRIPTION VARCHAR2 (400 BYTE)
RATING VARCHAR2 (4 BYTE)
CATEGORY VARCHAR2 (20 BYTE)
* RELEASE_DATE DATE
OMR_TITLEID_PK (TITLE_ID)
OMR_TITLEID_PK (TITLE_ID)
FLIX.RENTAL_HISTORY
PF* MEDIA_ID NUMBER (10)
P * RENTAL_DATE DATE
PF* CUSTOMER_ID NUMBER (10)
RETURN_DATE DATE
OMR_RENTALHISTORY_PK (CUSTOMER_ID, MEDIA_ID, RENTAL_DATE)
OMR_RENTALHISTORY_MEDIAID_FK (MEDIA_ID)
SYS_C008987 (CUSTOMER_ID)
OMR_RENTALHISTORY_PK (CUSTOMER_ID, MEDIA_ID, RENTAL_DATE)
FLIX.ACTORS
P * ACTOR_ID NUMBER (10)
* STAGE_NAME VARCHAR2 (40 BYTE)
* LAST_NAME VARCHAR2 (25 BYTE)
* FIRST_NAME VARCHAR2 (25 BYTE)
* BIRTH_DATE DATE
OMR_ACTORID_PK (ACTOR_ID)
OMR_ACTORID_PK (ACTOR_ID)
FLIX.MEDIA
P * MEDIA_ID NUMBER (10)
* FORMAT VARCHAR2 (3 BYTE)
F * TITLE_ID NUMBER (10)
OMR_MEDIAID_PK (MEDIA_ID)
OMR_MEDIA_TITLEID_FK (TITLE_ID)
OMR_MEDIAID_PK (MEDIA_ID)
FLIX.CUSTOMERS
P * CUSTOMER_ID NUMBER (10)
* LAST_NAME VARCHAR2 (25 BYTE)
* FIRST_NAME VARCHAR2 (25 BYTE)
* HOME_PHONE VARCHAR2 (12 BYTE)
* ADDRESS VARCHAR2 (100 BYTE)
* CITY VARCHAR2 (30 BYTE)
* STATE VARCHAR2 (2 BYTE)
EMAIL VARCHAR2 (25 BYTE)
CELL_PHONE VARCHAR2 (12 BYTE)
OMR_CUSTOMERID_PK (CUSTOMER_ID)
OMR_CUSTOMERID_PK (CUSTOMER_ID)

Create user

The user that would control the database would be Flix, which was first eliminated in case one already existed before, then it was created and given the respective permissions to manage the database but without altering the main system.

Tables and sequences

For the creation of the tables we first analyzed which tables do not depend on other tables and created those first, since to create a composite table we need other tables. Therefore, the Actors, Movies and Customers tables were created first. Then, the Star Billings table was created and finally the Media and Rental History tables were created. Each table was created with its respective constraints, primary keys and foreign keys so that they could be linked correctly.

In this case, sequences were created for the ID of each table that needed it, for example, this is the sequence for the ID of the Actors table, which started from 1001.

```
CREATE SEQUENCE actor_id_seq INCREMENT BY 1 START WITH 1001 NOCYCLE;
```

Insert data

When inserting data, CONSTRAINTS were introduced in the creation of tables, which means that in intermediate tables they are not allowed to insert data, since a movie could not be inserted with a media_id that does not exist.

That is why in tables such as customers, movies and actors which are catalog tables, it is for this reason that in these it is easier to insert data, since in these tables they do not have a number of CONSTRAINTS that can limit the insertion of data, in exchange for rental_history, media and star_billings, which are intermediate tables that need the catalog tables to be able to insert data into them.

To use the sequence of each ID when inserting data, it was written as SequenceName.nextval, as shown in the following example.

```
insert into actors values (actor_id_seq.NEXTVAL,'Tom
Hardy','Edward','Hardy','15-SEP-1977');
```

View, synonym and index

A view of the copies of unavailable movies called Title Unavailable, in other words, the movies that were rented and have not been returned, was created. As Oracle allows the creation of views, it could be done in an easy way and with the union of two tables.

Oracle also allows the creation of synonyms, which is the way to call an object but with another name, in this way the synonym for the Title Unavailable view was created.

The main purpose of an index is to optimize information filtering and searching, through the key fields in a table. Oracle allows the creation of indexes easily, in this case, the index for the "last name" column of the Customers table was created.

Queries

1) Show information of movies from six different tables

```
SELECT c.last_name, rh.rental_date, me.format, m.title, sb.comments,
a.stage_name
FROM customers c, rental_history rh, media me, movies m,
star_billings sb, actors a
WHERE c.customer_id = rh.customer_id
and rh.media_id = me.media_id
and me.title_id = m.title_id
and m.title_id = sb.title_id
and sb.actor_id = a.actor_id;
```

This is the same query as the previous one, but using inner join.

```
SELECT c.LAST_NAME, r.RENTAL_DATE, m.FORMAT, mo.TITLE, s.COMMENTS,
a.STAGE_NAME
from customers c inner join rental_history r on (c.CUSTOMER_ID =
r.CUSTOMER_ID)
INNER join media m on (r.MEDIA_ID = m.MEDIA_ID)
INNER JOIN movies mo on (m.TITLE_ID = mo.TITLE_ID)
INNER JOIN star_billings s on (mo.TITLE_ID = s.TITLE_ID)
INNER JOIN actors a on (s.ACTOR_ID = a.ACTOR_ID);
```

2) Show only of movies that have not been returned and that are rated R.

```
SELECT c.last_name, a.stage_name, m.title, m.rating, rh.return_date
FROM actors a, movies m, rental_history rh, star_billings sb, media
me, customers c
WHERE a.actor_id = sb.actor_id
      and m.title_id = sb.title_id
      and me.media_id = rh.media_id
      and m.title_id = me.title_id
      and c.customer_id = rh.customer_id
      and m.rating = 'R'
      and rh.return_date is NULL;
```

3) Display title, quantity of vhs and DVD per movie

```
select mo.title,
count(case me.format when 'DVD' then 'DVD'  end) "DVD",
count(case me.format when 'VHS' then 'VHS'  end) "VHS",
count(*) total
from movies mo INNER JOIN  media me on mo.title_id=me.title_id
group by mo.title;
```

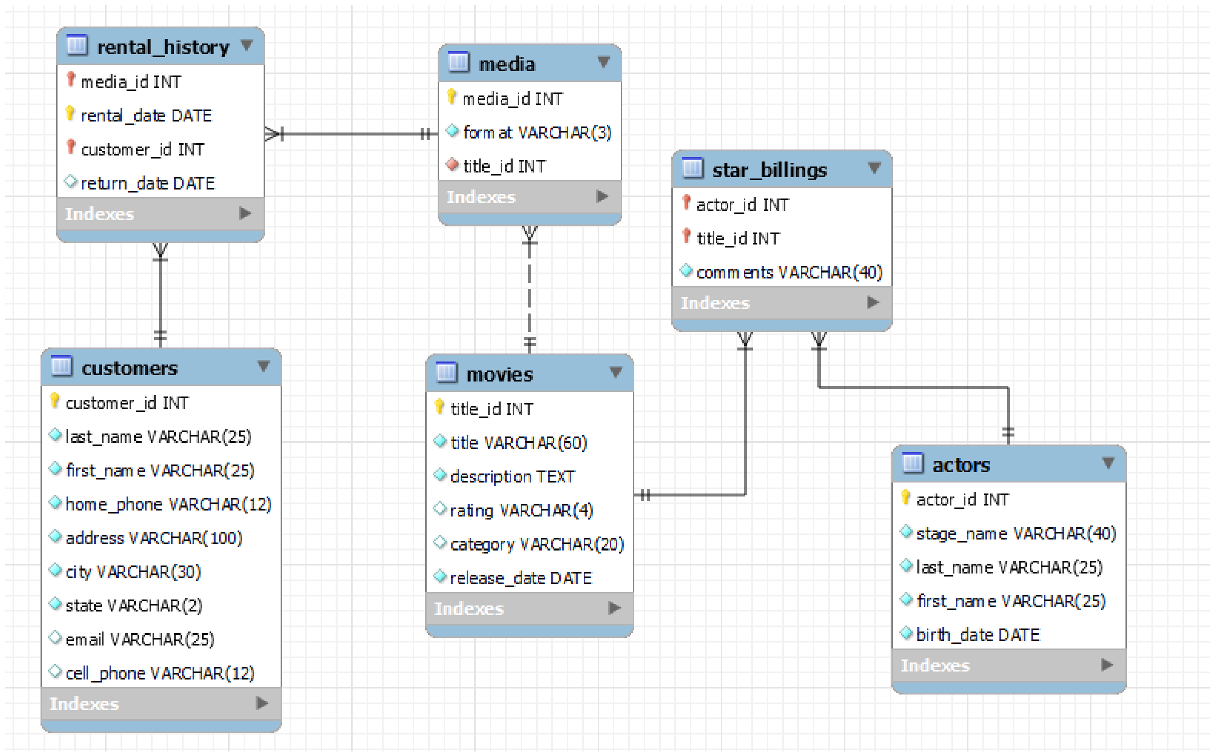
4) Show movie ID, movie name and number of DVD copies that took more than 15 days to be returned.

```
select a.title_id, d.titulo, d.Cantidad
from   movies   a   inner   join   (select   m.title   as   titulo,
COUNT(d.format) as Cantidad
from rental_history h inner join media d using (media_id)
                       inner join movies m using (title_id)
where      ((h.return_date-h.rental_date      >=      15)      or
(sysdate-h.rental_date >= 15))and d.format = 'DVD'
GROUP BY m.title) d on (a.title = d.titulo);
```

This is the same query, but using “with”.

```
with copies as
(SELECT e.media_id, e.title_id, m.title, e.format
from movies m, media e
where m.title_id = e.title_id
and e.format like 'DVD'),
      rentalday as
    (select media_id
    from rental_history
    where (return_date-rental_date)>15),
    rentaldelays as(select c.title_id, count(c.title_id) quantity
                    from copies c, rentalday r
                    where c.media_id=r.media_id
                    group by c.title_id)
      select rd.title_id, m.title, rd.quantity
    from movies m, rentaldelays rd
    where m.title_id = rd.title_id;
```

MySQL



Create User

To create a user we must create the database as long as it is not previously created, the user must be created along with its password in this case to avoid problems we use Flix as username and password, then we must grant permissions to the user according to the actions to be carried out.

Create Tables

Catalog or primary tables

When creating the tables it is essential to consider the order in which it is done, first creating those tables that have no dependency on any other.

- Movies
- Actors
- Customers

Movement or Secondary Tables

After having all the catalog tables, the intermediate tables, or movement tables, are created, which have fields that depend on the catalog tables that help to relate them (foreign keys).

- Star_billings (dependent on the actors and movies table)
- Rental_history (dependent on customer table)

- Media (depending on the movies table)

In MySQL the sequences do not exist, that is why you only change the number you want to start with and then insert data without the need to put the primary key field.

```
CREATE TABLE movies
    (title_id INT PRIMARY KEY AUTO_INCREMENT, …);

ALTER TABLE Movies AUTO_INCREMENT = 1;
```

Insert data

When inserting data, in the MySQL database, in the use of this database in the customers, media, actors and movies tables, they handle an auto incrementing field in Oracle, the sequence is called and it is told to increment the next number of the sequence, in MySQL since there are no sequences, the table must be altered so that they start with a desired number, and when inserting data, it is not necessary to put the auto incrementing column, since in MySQL when declaring auto incrementing, when inserting values, the auto incrementing takes the next value.

```
insert into media (format,title_id) values ('DVD','1');
```

The mean table uses a sequence, the columns of the table are entered omitting the autoincrement column and the values are entered.

View, synonym and index

In MySQL the same view was created, which consists of the copies of unavailable movies, however the syntax in MySQL changes a little, but this was not an impediment to perform it.

A view is created from a query that returns the movies that have not been returned.

```
create view title_unavail
AS
Select movies.title_id, media.media_id, rental_history.rental_date
from Online_Media_Rental.movies
,Online_Media_Rental.rental_history,Online_Media_Rental.media
where rental_history.media_id = media.media_id
and movies.title_id = media.title_id
and return_date is null;
```

The index was created the same way as in oracle for the column “Last Name”.

In MySql there are no synonyms as such, an alternative is to create another view with the alternative name that we want to assign to the original view with all the information with it.

```
create view tu as
select * from title_unavail;

select * from tu;
```

Trying to increment a synonym for the newly created view, we came across the detail that MySQL does not have the option to create synonyms.
Note : "*Synonyms are database dependent and cannot be accessed by other databases" (SQL | SYNONYM - IBM, 2020)"*

Queries

All the queries mentioned before in the Oracle section work the same way in MySQL, so there was no need to make any changes and the results shown by the queries were as expected.

Conclusion

The development of this database implied having knowledge about the links between tables by means of primary keys, foreign keys and constraints. Thanks to this, it was possible to control the data in the database and avoid errors between relationships.

Having developed this database in two different platforms showed that there is no problem using different platforms as long as you have the base structure of what will be done. In addition, working as a team in this project was essential to detect failures and changes that existed between Oracle and MySQL, to correct them later and efficiently complement all the parts of the database.