

# Base de Datos Avanzada

---

Edher Nuño

Miguel Ochoa

18 de agosto de 2018

Universidad Autónoma de Guadalajara

# Problemática

---

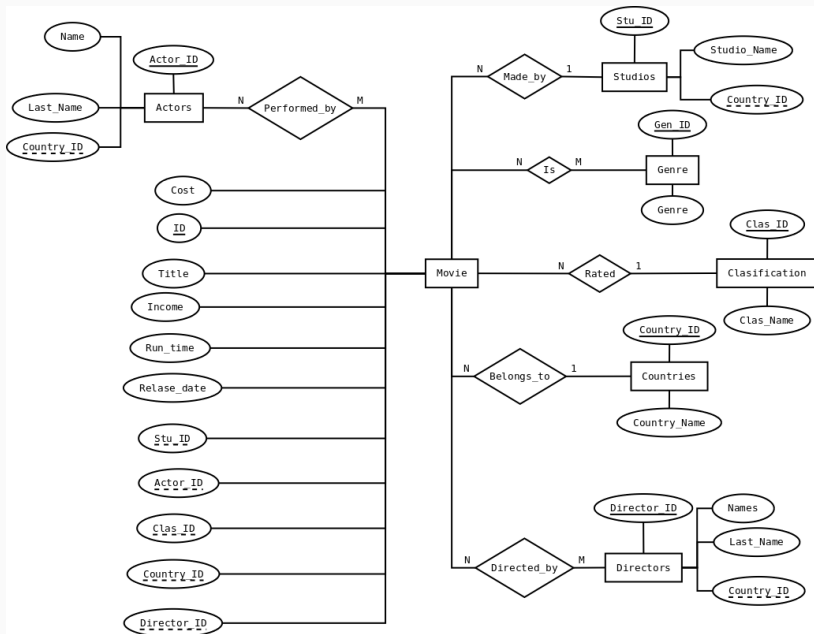
Colección de películas con :

- Entradas repetidas sin control.
- Pocas a nulas referencias para intercambio.
  - a) Estudio.
  - b) Director.
- Información de parametros.
  - a) Duración.
  - b) Budget vs Revenue.

## **Diseño de la base de datos.**

---

# Diagrama Entidad-Relación



# Implementación

---

# Implementación

```
CREATE TABLE  
COUNTRY (ID INTEGER  
NOT NULL GENERATED  
ALWAYS AS IDENTITY  
(START WITH 1  
INCREMENT BY 1),  
NAME VARCHAR(30)  
UNIQUE NOT NULL,  
PRIMARY KEY(ID) );
```

Crea la tabla de países con el campo ID como llave primaria y generada automáticamente

## db2 LIST TABLES FOR SCHEMA MOVIEINDEX

Table/View	Schema	Type	Creation time
ACTOR	MOVIEINDEX	T	2018-08-08-14.44.32.714018
CLASIFICATION	MOVIEINDEX	T	2018-08-08-14.44.32.933873
COUNTRY	MOVIEINDEX	T	2018-08-08-14.44.32.083194
DIRECTOR	MOVIEINDEX	T	2018-08-08-14.44.32.505958
GENDERS	MOVIEINDEX	T	2018-08-08-14.44.32.312163
MOVIE	MOVIEINDEX	T	2018-08-08-15.15.57.504164
STUDIO	MOVIEINDEX	T	2018-08-08-15.15.57.273201

7 record(s) selected.



db2 DESCRIBE TABLE MOVIEINDEX.MOVIE

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
ID	SYSIBM	INTEGER	4	0	No
NAME	SYSIBM	VARCHAR	50	0	No
DURATION	SYSIBM	INTEGER	4	0	No
COST	SYSIBM	DOUBLE	8	0	No
INCOME	SYSIBM	DOUBLE	8	0	No
RELEASE_DATE	SYSIBM	DATE	4	0	No
CLASIFICATIONID	SYSIBM	INTEGER	4	0	No
COUNTRYID	SYSIBM	INTEGER	4	0	No
STUDIOID	SYSIBM	INTEGER	4	0	No
DIRECTORID	SYSIBM	INTEGER	4	0	No
10 record(s) selected.					

# Tablas

ID	NAME
1	France
2	Mexico
3	Germany
4	China
5	Cuba
6	Rusia
7	Sweden
8	Canada
9	England
10	Poland
11	Spain
12	Australia
13	New Zealand
14	Japan
15	USA
16	Hungary
17	Austria

17 record(s) selected.

(a) Country Table

ID	NAME	NATIONALITYID
1	Universal	15
2	FOX	15
3	Tequila	2
4	Vodka Studio	6
5	Sorry Studio	8
6	Klavis Studio	13
7	Copys	4
8	Disney	15
9	Pixar	15
10	Dream Works	15

10 record(s) selected.

(b) Studio Table

ID	NAME	LASTNAME	NATIONALITYID
1	Steven	Spielberg	15
2	Tod	Browning	15
3	Ishirou	Honda	14
4	Michael	Curtiz	16
5	Robert	Wise	15
6	Ridley	Scott	15
7	Michael	Haneke	17
8	Akira	Kurosawa	14
9	George	A. Romero	15
10	Robert	Wiene	10

10 record(s) selected.

(c) Director Table

**Figura 1:** Tablas en la MOVIESDB parte 1

# Tablas

ID	NAME
1	Adventure
2	Terror
3	Drama
4	Action
5	Comedy
6	Science fiction
7	Romance
8	Animation
9	Suspense
10	Art
11	Documentary

11 record(s) selected.

(a) Gender Table

ID	NAME	LASTNAME	NATIONALITY
1	Sam	Neill	15
2	Laura	Dern	15
3	Jeff	Goldblum	15
4	Bela	Lugosi	16
5	Dwight	Frye	15
6	Haruo	Nakajima	14
7	Akira	Takarada	14
8	Humphrey	Bogart	15
9	Ingrid	Bergman	7
10	Michael	Rennet	9
11	Sigourney	Weaver	15
12	Isabelle	Huppert	1
13	Toshiro	Mifune	14
14	Machiko	Kyou	14
15	Masayuki	Mori	14
16	Duane	Jones	15
17	Judith	Odeh	15
18	Werner	Krauss	3
19	Conrad	Veidt	3

19 record(s) selected.

(b) Actor Table

ID	SHORT_NAME	NAME
1	G	General Audiences
2	PG	Parental Guidance Suggested
3	PG-13	Parents Strongly Cautioned
4	R	Restricted
5	NC-17	Adults Only

5 record(s) selected.

(c) Clasification  
Table

Figura 2: Tablas en la MOVIESDB parte 2

# Vistas

---

# Vistas

Country	Movie Name	Relase Date
USA	Jurassic Park	1993-06-09
Japan	Rashomon	1950-08-25
. . .	. . .	. . .

**(a)** Vista: Peliculas por paises

Name	Duration	Cost	Income	Release Date	Clasification	Studio	Director
Jurassic Park	127	\$63 Mi-llones	\$920.1 Millones	1993-06-09	PG-13	Universal	Steven Spielberg
Jurassic Park	127	\$63 Mi-llones	\$920.1 Millones	1993-06-09	PG-13	Universal	Steven Spielberg
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

**(b)** vista1

Para esta Base de Datos se implementaron 5 *Vistas*:

- *COUNTRYID\_MOVIES* : Permite ver las entradas de la tabla película ordenadas en base al país de origen (alfabéticamente). El resultado de esta *vista* se presenta en ejemplo en la figura 4.
- *TOP\_INCOME\_MOVIES* : Permite ver una lista de las 10 películas con mayores ingresos.
- *LOWEST\_INCOME\_MOVIES* : Permite ver una lista de las 10 películas con menores ingresos.

- *TOP\_EXPENSIVE\_MOVIES*: Muestra todas las entradas de la tabla *MOVIE* ordenadas por su costo.
- *MOVIES\_REVENUES* : Muestra una lista de las películas con mayores ganancias, un ejemplo de esta vista se muestra en la figura 6.

COUNTRY	MOVIENAME	RELEASE_DATE
France	La Pianiste	03/28/2002
Germany	Das Cabinet des Dr. Caligari	03/19/1921
Japan	Godzilla	12/03/1954
Japan	Rashomon	08/25/1950
USA	Jurassic Park	06/09/1993
USA	Night of the Living Dead	10/01/1968
USA	Alien	05/25/1979
USA	The Day the Earth Stood Still	09/17/1951
USA	Dracula	02/12/1931

**Figura 4:** Vista de películas en base a su país de origen.



NAME	DIRECTORID	DURATION	COST	INCOME	RELEASE_DATE	CLASSIFICATION	STUDIO
Jurassic Park	Steven	127	+6.300000000000000E+007	+9.201000000000000E+008	06/09/1993	Parents Strongly Cautioned	Universal
Dracula	Tod	72	+3.550000000000000E+005	+3.550000000000000E+005	02/12/1931	Parents Strongly Cautioned	Universal
Godzilla	Ishiro	96	+1.000000000000000E+006	+1.500000000000000E+006	12/03/1954	Parents Strongly Cautioned	Toho
The Day the Earth Stood Still	Robert	92	+1.200000000000000E+006	+1.200000000000000E+006	09/17/1951	Parents Strongly Cautioned	FOX
Alien	Ridley	117	+1.100000000000000E+007	+1.049310010000000E+008	05/25/1979	Restricted	FOX
La Planète	Michael	131	+3.000000000000000E+006	+9.001375000000000E+006	03/28/2002	Restricted	Arte France Cinéma
Rashomon	Akira	88	+2.500000000000000E+005	+9.650000000000000E+004	08/25/1950	Parents Strongly Cautioned	Toho
Night of the Living Dead	George	96	+1.140000000000000E+005	+3.000000000000000E+007	10/01/1968	Restricted	Image Ten
Das Cabinet des Dr. Caligari	Robert	78	+1.000000000000000E+004	+1.800000000000000E+004	03/19/1921	Restricted	Decca-Bioscop

**Figura 5:** Vista de películas en base sus Recaudaciones.

NAME	DURATION	COST	INCOME	RELEASE_DATE	CLASSIFICATION	STUDIO	DIRECTOR
Jurassic Park	127	+0.3000000000000000E+007	+9.201000000000000E+008	06/09/1993	Parents Strongly Cautioned	Universal	Steven
Dracula	72	+3.550000000000000E+005	+3.550000000000000E+005	02/12/1931	Parents Strongly Cautioned	Universal	Tod
Godzilla	96	+1.000000000000000E+006	+1.500000000000000E+006	12/03/1954	Parents Strongly Cautioned	Toho	Ishiro
The Day the Earth Stood Still	92	+1.200000000000000E+006	+1.200000000000000E+006	09/17/1951	Parents Strongly Cautioned	FOX	Robert
Alien	117	+1.100000000000000E+007	+1.049310010000000E+008	05/25/1979	Restricted	FOX	Ridley
La Planète	131	+3.000000000000000E+006	+9.001375000000000E+006	03/28/2002	Restricted	Arte France cinéma	Michael
Rashomon	88	+2.500000000000000E+005	+9.050000000000000E+004	08/25/1950	Parents Strongly Cautioned	Toho	Akira
Night of the Living Dead	96	+1.140000000000000E+005	+3.000000000000000E+007	10/01/1968	Restricted	Image Ten	George
Das Cabinet des Dr. Caligari	78	+1.800000000000000E+004	+1.800000000000000E+004	03/19/1921	Restricted	Decla-Bioscop	Robert

**Figura 6:** Vista de películas en base sus Ganancias.

# Index

---

Para agilizar el resultado de las peticiones a las vistas mencionadas se implementaron los siguientes *Index*:

```
CREATE INDEX COUNTRY_ID  ON MOVIEINDEX.COUNTRY(NAME);  
CREATE INDEX INCOME     ON MOVIEINDEX.MOVIE(INCOME);  
CREATE INDEX COST       ON MOVIEINDEX.MOVIE(COST);  
CREATE INDEX MOVIE_INDEX ON MOVIEINDEX.MOVIE(NAME);
```

# Store Procedures

---

## Store Procedures

Se utilizaron 5 procedimientos almacenados:

1. Un procedimiento carga los datos a la tabla que vincula a los actores a cada película.
2. Un procedimiento carga los datos a la tabla que vincula a los géneros con cada película.
3. Un procedimiento genera una lista de las películas con sus géneros.
4. Un procedimiento genera una lista de las películas con sus actores.
5. Un procedimiento para calcular las ganancias de las películas.

# Store Procedures

## Procedimiento 1:

```
CREATE or REPLACE PROCEDURE MOVIEINDEX.MOVIE_ACTOR ( IN MOVIE_NAME VARCHAR(50), IN ACTOR_NAME VARCHAR(50) )
LANGUAGE SQL
P1:BEGIN
    DECLARE ACTOR_ID int;
    DECLARE MOVIE_ID int;

    DECLARE cursor1 CURSOR FOR SELECT ID FROM MOVIEINDEX.MOVIE WHERE MOVIE_NAME = MOVIEINDEX.MOVIE.NAME;
    DECLARE cursor2 CURSOR FOR SELECT ID FROM MOVIEINDEX.ACTOR WHERE ACTOR_NAME = MOVIEINDEX.ACTOR.NAME;
    OPEN cursor1;
        FETCH FROM cursor1 INTO ACTOR_ID;
    CLOSE cursor1;
    OPEN cursor2;
        FETCH FROM cursor2 INTO MOVIE_ID;
    CLOSE cursor2;

    INSERT INTO MOVIEINDEX.ACTOR_MOVIE (ACTOR_ID, MOVIE_ID) VALUES (ACTOR_ID, MOVIE_ID);
END P1
@
```

## Procedimiento 2:

```
CREATE or REPLACE PROCEDURE MOVIEINDEX.MOVIE_GENDER ( IN MOVIE_NAME VARCHAR(50), IN GENDER_NAME VARCHAR(50)
LANGUAGE SQL
P1:BEGIN
    DECLARE GENDER_ID int;
    DECLARE MOVIE_ID int;

    DECLARE cursor1 CURSOR FOR SELECT ID FROM MOVIEINDEX.MOVIE WHERE MOVIE_NAME = MOVIEINDEX.MOVIE.NAME;
    DECLARE cursor2 CURSOR FOR SELECT ID FROM MOVIEINDEX.GENDER WHERE GENDER_NAME = MOVIEINDEX.GENDER.NAME;
    OPEN cursor1;
        FETCH FROM cursor1 INTO GENDER_ID;
    CLOSE cursor1;
    OPEN cursor2;
        FETCH FROM cursor2 INTO MOVIE_ID;
    CLOSE cursor2;

    INSERT INTO MOVIEINDEX.GENDER_MOVIE (GENDERID, MOVIEID) VALUES (GENDER_ID, MOVIE_ID);
END P1
@
```



## Procedimiento 3:

```
CREATE or REPLACE PROCEDURE MOVIEINDEX.GENDERS_PER_MOVIE ( IN MOVIEName VARCHAR(50), OUT GENDERS INT )  
LANGUAGE SQL  
P5:BEGIN  
    DECLARE MOVIEID int;  
  
    DECLARE cursor1 CURSOR FOR SELECT ID FROM MOVIEINDEX.MOVIE WHERE MOVIEINDEX.MOVIE.NAME = MOVIEName;  
    OPEN cursor1;  
        FETCH FROM cursor1 INTO MOVIEID;  
    CLOSE cursor1;  
  
    DECLARE cursor2 CURSOR FOR SELECT COUNT(*) AS GENDERS_IN FROM MOVIEINDEX.MOVIE,  
    MOVIEINDEX.GENDER_MOVIE, MOVIEINDEX.GENDER  
    WHERE MOVIEINDEX.MOVIE.ID = MOVIEID AND  
    MOVIEINDEX.GENDER_MOVIE.MOVIEDID = MOVIEID AND  
    MOVIEINDEX.GENDER.ID = MOVIEINDEX.GENDER_MOVIE.GENDERID;  
    OPEN cursor2;  
        FETCH FROM cursor2 INTO GENDERS;  
    CLOSE cursor2;  
END P5  
@
```

## Procedimiento 4:

```
CREATE or REPLACE PROCEDURE MOVIEINDEX.ACTOR_IN_MOVIE ( IN MOVIEName VARCHAR(50), OUT ACTORS INT )
LANGUAGE SQL
P4:BEGIN
    DECLARE MOVIEID int;

    DECLARE cursor1 CURSOR FOR SELECT ID FROM MOVIEINDEX.MOVIE WHERE MOVIEINDEX.MOVIE.NAME = MOVIEName;
    OPEN cursor1;
        FETCH FROM cursor1 INTO MOVIEID;
    CLOSE cursor1;

    DECLARE cursor2 CURSOR FOR SELECT COUNT(*) AS ACTORS_IN FROM MOVIEINDEX.MOVIE,
    MOVIEINDEX.ACTOR_MOVIE,
    MOVIEINDEX.ACTOR WHERE MOVIEINDEX.MOVIE.ID = MOVIEID AND
    MOVIEINDEX.ACTOR_MOVIE.MOVIEDID = MOVIEID AND
    MOVIEINDEX.ACTOR.ID = MOVIEINDEX.ACTOR_MOVIE.ACTORID;
    OPEN cursor2;
        FETCH FROM cursor2 INTO ACTORS;
    CLOSE cursor2;
END P4
@
```

## Procedimiento 5:

```
CREATE or REPLACE PROCEDURE MOVIEINDEX.REVENUE ( IN MOVIEName VARCHAR(50), OUT TOTAL FLOAT )  
LANGUAGE SQL  
P1:BEGIN  
  
    DECLARE cursor1 CURSOR FOR SELECT SUM(INCOME - COST) AS MOVIEREVENUE FROM MOVIEINDEX.MOVIE WHERE MOVIEName = MOVIEName;  
    OPEN cursor1;  
        FETCH FROM cursor1 INTO TOTAL;  
    CLOSE cursor1;  
END P1  
@
```

# Triggers

---

Los 5 *Triggers* considerados son:

1. Validación de dato en actualizaciones para el campo *INCOME* en la tabla *MOVIE*.
2. Validación de dato en inserciones para el campo *INCOME* en la tabla *MOVIE*.
3. Validación de dato en inserciones para el campo *COST* en la tabla *MOVIE*.
4. Validación de dato en actualizaciones para el campo *COST* en la tabla *MOVIE*.
5. Validación de dato en inserciones para el campo *DURATION* en la tabla *MOVIE*.

## Trigger 1:

```
CREATE OR REPLACE TRIGGER INCOME_MOVIE
  BEFORE UPDATE OF INCOME ON MOVIEINDEX.MOVIE
  REFERENCING OLD AS OLD_MOVIE
               NEW AS NEW_MOVIE
  FOR EACH ROW MODE DB2SQL
  WHEN (NEW_MOVIE.INCOME < OLD_MOVIE.INCOME )
  BEGIN ATOMIC
    SIGNAL SQLSTATE '75001' ('INVALID INCOME, IT MOST BE HIGHER THAN THE LAST ONE')
  END;
```

## Trigger 2:

```
CREATE OR REPLACE TRIGGER VALID_INCOME
  BEFORE INSERT ON MOVIEINDEX.MOVIE
  REFERENCING NEW AS NEW_MOVIE
  FOR EACH ROW MODE DB2SQL
  WHEN (NEW_MOVIE.INCOME <= 0)
  BEGIN ATOMIC
    SIGNAL SQLSTATE '75001' ('INCOME MUST BE MORE THAN 0')
  END;
```

## Trigger 3:

```
CREATE OR REPLACE TRIGGER VALID_COST
  BEFORE INSERT ON MOVIEINDEX.MOVIE
  REFERENCING NEW AS NEW_MOVIE
  FOR EACH ROW MODE DB2SQL
  WHEN (NEW_MOVIE.COST <= 0)
  BEGIN ATOMIC
    SIGNAL SQLSTATE '75001' ('COST MUST BE MORE THAN 0')
  END;
```



## Trigger 4:

```
CREATE OR REPLACE TRIGGER COST_MOVIE
  BEFORE UPDATE OF COST ON MOVIEINDEX.MOVIE
  REFERENCING OLD AS OLD_MOVIE
               NEW AS NEW_MOVIE
  FOR EACH ROW MODE DB2SQL
  WHEN (NEW_MOVIE.COST < OLD_MOVIE.COST )
  BEGIN ATOMIC
    SIGNAL SQLSTATE '75001' ('INVALID COST, IT MOST BE HIGHER THAN THE LAST ONE')
  END;
```

## Trigger 5:

```
CREATE OR REPLACE TRIGGER VALID_DURATION
  BEFORE INSERT ON MOVIEINDEX.MOVIE
  FOR EACH ROW MODE DB2SQL
  WHEN (MOVIE.DURATION <= 0)
  BEGIN ATOMIC
    SIGNAL SQLSTATE '75001' ('DURATION MUST BE MORE THAN 0')
  END;
```