Unidad 4: Modelo físico. SQL

Práctica 4. Resolver en MySQL.

En todos los casos, se solicita escribir la sentencia correspondiente y, a continuación, ejecutarla para verificar el resultado.

Si tras la sentencia, escribimos DESC NbreTabla, MySQL nos mostrará la estructura de la tabla resultante. Ejemplo: DESC Peliculas;

Creación de tablas. Modificación de la estructura de la tabla.

1. Crear una base de datos llamada Peliculas + 'tu apellido'.
2. Dentro de esta BD, crear una tabla tabla "Actores" que contenga las columnas: IdActor (int), AUTOINCREMENTAL, NombreActor, ApellidoActor, ambos nvarchar(50) y GeneroActor como nvarchar(1), FechaNacActor (DATE), FechaFallecActor (DATE); IdActor será clave principal.
3. Escribir una sentencia SQL para crear una tabla "Directores" que contenga las columnas: (PK) IdDirector (Int), NombreDirector y ApellidoDirector, ambos nvarchar(45), comprobando si la tabla existe (cláusula IF NOT EXISTS).
4. Modificar la estructura de la tabla "Actores" de modo tal que el valor de la clave principal NO sea autoincremental.
5. Escribir una sentencia SQL para crear una tabla "actores_dup", cuya estructura sea copia de la estructura de la tabla "actores". (cláusula LIKE):

   CREATE TABLE actores_dup LIKE actores;

6. Crear una tabla "direccion" con dos campos int: IdDirector e IdPelicula.

```
CREATE TABLE direccion (
IdDirector INT NOT NULL,
IdPelicula INT NOT NULL)
```

7. Modificar la tabla "Dirección", de modo tal que la clave principal quede formada por ambos campos, IdDirector e IdPelicula; clave que debe ser única

```
ALTER TABLE direccion
CHANGE COLUMN  idDirector INT NOT NULL ,
CHANGE COLUMN idPelicula INT NOT NULL ,
ADD PRIMARY KEY (idDirector, idPelicula);
;
```

   a. ¿Qué significa esto último? IdDirector e IdPelicula son la clave principal.
   b. ¿Pudiste resolverlo con una única sentencia?
8. A continuación, escribir las sentencias para crear las siguientes tablas; en todos los casos, los campos subrayados son PK:
   a. Reparto: IdActor Int, IdPelicula Int, Rol nvarchar(45)

```
CREATE TABLE reparto (
```

IdActor INT NOT NULL,

IdPelicula INT NOT NULL,

Rol VARCHAR(45) NULL,

PRIMARY KEY (IdActor, IdPelicula));

b. Críticos: IdCritico INT, NombreCritico nvarchar(45)

```
CREATE TABLE criticos (
 IdCritico`INT NOT NULL,
  NombreCritico VARCHAR(45) NULL,
 PRIMARY KEY (IdCritico));
```

c. Genero: IdGenero INT, Genero nvarchar(20)

```
CREATE TABLE genero (
IdGenero INT NOT NULL,
Genero VARCHAR(20) NULL,
PRIMARY KEY (IdGenero));
```

d. GenerosPeliculas: IdPelicula Int, IdGenero INT

```
CREATE TABLE generospeliculas (
 IdPelicula INT NOT NULL,
 IdGenero INT NOT NULL,
 PRIMARY KEY (IdPelicula,  IdGenero));
```

e. Ranqueo: IdPelicula INT, IdCritico INT, estrellas decimal (3,2), puntos int;

```
CREATE TABLE ranqueo (
 IdPelicula INT NOT NULL,
 IdCritico INT NULL,
 Estrellas DECIMAL(3,2) NULL,
 Puntos INT NULL,
 PRIMARY KEY (IdPelicula));
```

f. Peliculas: IdPelicula (INT), Titulo (100), AnioPelicula Int, Duracion INT, Idioma nvarchar (50), fechaLanzamiento Date , PaisRealizacion char(5)

```
CREATE TABLE peliculas (
 idPelicula INT NULL,
 Titulo VARCHAR(100) NULL,
 AnioPelicula INT NULL,
 Duracion INT NULL,
 Idioma VARCHAR(50) NULL,
 FechaLanzamiento DATE NULL,
```

```
`PaisRealizacion CHAR(5) NULL,
 PRIMARY KEY (idPelicula));
```

9. Modificar el nombre de la tabla Ranqueo por Puntaje.

```
ALTER TABLE ranqueo
RENAME TO  puntaje ;
```

10. Modificar la estructura de la tabla “Peliculas” de modo tal que ningún campo sea nulo, a excepción del campo FechaLanzamiento, y cambiar el nombre del campo “Titulo” por NombrePelicula.

```
ALTER TABLE peliculas
CHANGE COLUMN NombrePelicula VARCHAR(100) NOT NULL ,
CHANGE COLUMN  AnioPelicula INT NOT NULL ,
CHANGE COLUMN Duracion INT NOT NULL ,
CHANGE COLUMN Idioma VARCHAR(50) NOT NULL ,
CHANGE COLUMN PaisRealizacion CHAR(5) NOT NULL ;
```

Inserción de valores (tuplas)

1. Escribir una sentencia, o grupo de sentencias, que permita ingresar los siguientes registros en la tabla Criticos.

```
ID        |       NbreCritico
--------+-------------------------------
  9001 | Righty Sock
  9002 | Jack Malvern
  9003 | Flagrant Baronessa
  9004 | Alec Shaw
  9005 |
  9006 | Victor Woeltjen
  9007 | Simon Wright
  9008 | Neal Wruck
  9009 | Paul Monks
  9010 | Mike Salvati
  9011 |
  9012 | Wesley S. Walker
  9013 | Sasha Goldshtein
  9014 | Josh Cates
  9015 | Krug Stillo
  9016 | Scott LeBrun
  9017 | Hannah Steele
```

9018 | Vincent Cadena
9019 | Brandt Sponseller
9020 | Richard Adams

INSERT INTO criticos (IdCritico, NombreCritico)
VALUES (9001, “Righty Sock”),
(9002, "Jack Malvern"),
(9003, "Flagrant Baronessa"),
(9004, "Alec Shaw"),
(9005, ""),
(9006, "Victor Woeltjen"),
(9007, "Simon Wright"),
(9008, "Neal Wruck"),
(9009, "Paul Monks"),
(9010, "Mike Salvati"),
(9011, ""),
(9012, "Wesley S. Walker"),
(9013, "Sasha Goldshtein"),
(9014, "Josh Cates"),
(9015, "Krug Stillo"),
(9016, "Scott LeBrun"),
(9017, "Hannah Steele"),
(9018, "Vincent Cadena"),
(9019, "Brandt Sponseller"),
(9020, "Richard Adams");

2. Escribir una sentencia SQL para ingresar, en la tabla actores, los siguientes datos (notar que la fecha se ingresa con formato AAAA-MM-DD, entre comillas simples):

101	James	Stewart	M	1908-05-20	1997-07-02
102	Deborah	Kerr	F	1921-09-30	2007-10-16
103	Peter	OToole	M	1932-07-02	2013-12-14
104	Robert	De Niro	M	1943-07-17	
105	F. Murray	Abraham	M	1931-10-24	
106	Harrison	Ford	M	1942-07-13	
107	Nicole	Kidman	F	1967-06-20	
108	Stephen	Baldwin	M	1966-05-12	
109	Jack	Nicholson	M	1937-04-22	
110	Mark	Wahlberg	M	1971-06-05	
111	Woody	Allen	M	1935-12-01	
112	Claire	Danes	F		
113	Tim	Robbins	M	1958-10-16	
114	Kevin	Spacey	M	1959-07-26	
115	Kate	Winslet	F	1975-10-05	
116	Robin	Williams	M	1951-07-21	2014-07-11

117	Jon	Voight	M	1938-12-29
118	Ewan	McGregor	M	1971-03-31
119	Christian	Bale	M	1974-01-30
120	Maggie	Gyllenhaal	F	1977-11-16
121	Dev	Patel	M	1990-04-23
122	Sigourney	Weaver	F	1949-10-08
123	David	Aston	M	
124	Ali	Astin	F	1966-11-27

INSERT INTO actores (idactor, NombreActor, ApellidoActor, GeneroActor, FechaNacActor, FechaFallecActor)
VALUES (101, "James", "Stewart", "M", '1908-05-20', '1997-07-02'),
(102, "Deborah", "Kerr", "F", '1921-09-30', '2007-10-16'),
(103, "Peter", "OToole", "M", '1932-07-02', '2013-12-14'),
(104, "Robert", "De Niro", "M", '1943-07-17', null),
(105, "F. Murray", "Abraham", "M", '1931-10-24', null),
(106, "Harrison", "Ford", "M", '1942-07-13', null),
(107, "Nicole", "Kidman", "F", 1967-06-20', null),
(108, "Stephen", "Baldwin", "M", '1966-05-12', null),
(109, "Jack", "Nicholson", "M", '1937-04-22', null),
(110, "Mark", "Wahlberg", "M", '1971-06-05', null),
(111, "Woody", "Allen", "M", '1935-12-01', null),
(112, "Claire", "Danes", "F", null, null),
(113, "Tim", "Robbins", "M", '1958-10-16', null),
(114, "Kevin", "Spacey", "M", '1959-07-26', null),
(115, "Kate", "Winslet", "F", '1975-10-05', null),
(116, "Robin", "Williams", "M", '1951-07-21', '2014-07-11'),
(117, "Jon", "Voight", "M", '1938-12-29', null),
(118, "Ewan", "McGregor", "M", '1971-03-31', null),
(119, "Christian", "Bale", "M", '1974-01-30', null),
(120, "Maggie", "Gylenhaal", "F", '1977-11-16', null),
(121, "Dev", "Patel", "M", '1990-04-23', null),
(122, "Sigourney", "Weaver", "F", '1949-10-08', null),
(123, "David", "Aston", "M", null, null),
(124, "Ali", "Astin", "F", '1966-11-27', null);

3. A continuación, vas a insertar los registros de las tablas restantes (pedir al docente).

Unidad 4: Modelo físico. SQL
Práctica 4. Resolver en MySQL.

En todos los casos, se solicita escribir la sentencia correspondiente y, a continuación, ejecutarla para verificar el resultado.
III. Consultas de selección

Escribir una consulta que devuelva el nombre y el año de cada película.
select NombrePelicula, AnioPelicula from peliculas;

Escribir una consulta que muestre la fecha en que fue estrenada American Beauty.
select AnioPelicula from peliculas where NombrePelicula = "American Beauty";

Escribir una consulta para obtener las películas que fueron lanzadas en 1999 (considerar el campo año).
select idPelicula, NombrePelicula from peliculas where AnioPelicula = 1999;

Escribir una consulta que devuelva las películas con su fecha de lanzamiento, ordenadas por nombre de película.
select NombrePelicula, AnioPelicula from peliculas
order by NombrePelicula;

Escribir una consulta para obtener las películas que fueron lanzadas con anterioridad a 1998; mostrarlas ordenadas por año de lanzamiento y alfabéticamente por nombre.
select AnioPelicula, NombrePelicula from peliculas where AnioPelicula < 1998
order by AnioPelicula asc, NombrePelicula asc;

Escribir una consulta para listar todos los actores y actrices que hayan fallecido (condición IS NULL).
select NombreActor, ApellidoActor, FechaFallecActor from actores where FechaFallecActor is not null;

Ídem anterior pero mostrando apellido y nombre concatenado de la siguiente manera: Apellido, Nombre [Función CONCAT()].
select CONCAT(ApellidoActor, “ “, NombreActor) from actores where FechaFallecActor is not null;

Unidad 4: Modelo físico. SQL

Práctica 4.

Te sugiero, escribir las sentencias en MySQL Workbench, comprobar el resultado conforme la consigna y, luego, pegar la instrucción debajo de cada ítem, según corresponda. Cualquier observación o aclaración que necesites hacer, podés añadirla al documento.

Resolver en MySQL.

IV. Funciones de agregación (SUM, MAX). Cláusula Group By.

1. Generar una consulta para obtener el ***menor*** tiempo de duración de una película (función MIN).

```
select min(Duracion) from peliculas;
```

2. Mostrar nombre de la película y duración, de la película con ***mayor*** duración (Función MAX). Ejecutar una consulta de selección para verificar el resultado.
    a. ¿qué ocurre en este caso? Devuelve correctamente la máxima duración pero el nombre de la película no se corresponde con la duración.

```
select NombrePelicula, max(Duracion) from peliculas
```

3. Mostrar la suma de puntos otorgada por cada crítico, agrupada por idcrítico.

```
select IdCritico, Puntos from puntaje

group by IdCritico;
```

4. Ídem anterior pero agrupado por idpelícula.

```
select IdPelicula, Puntos from puntaje

group by IdPelicula;
```

Unidad 4: Modelo físico. SQL

Práctica 4.

Te sugiero, escribir las sentencias en MySQL Workbench, comprobar el resultado conforme la consigna y, luego, pegar la instrucción debajo de cada ítem, según corresponda. Cualquier observación o aclaración que necesites hacer, podés añadirla al documento.

Resolver en MySQL.

V. Operadores lógicos AND y OR.

1. Escribir una consulta que devuelva el ID del actor cuyo nombre sea 'Woody' y su apellido, 'Allen'.

```
select idactor from actores where NombreActor = "Woody" and ApellidoActor = "Allen";
```

2. Escribir una consulta que devuelva los nombres de películas y fechas de lanzamiento, cuyo ID es: 905, 907, 917.

```
select NombrePelicula, FechaLanzamiento from peliculas

where (idPelicula = 905

or idPelicula = 907

or idPelicula = 917);
```

   a. Reescribir la instrucción utilizando el OPERADOR IN.

```
select NombrePelicula, FechaLanzamiento from peliculas

where idPelicula in (905, 907, 917);
```

   b. Comparar ambos resultados y establecer una valoración personal.

El uso del operador IN simplifica mucho la sentencia al permitir escribir los tres IDs a consultar en la misma línea.

3. Obtener las películas de la década del 60 (entre 1960 y 1969, incluidos) que hayan sido realizadas en el Reino Unido.

```
select NombrePelicula from peliculas

where AnioPelicula > 1960 and AnioPelicula < 1969 and PaisRealizacion = "UK";
```

   a. Reescribir la instrucción utilizando BETWEEN().

```
select NombrePelicula from peliculas

where AnioPelicula between 1960 and 1969 and PaisRealizacion = "UK";
```

   b. Comparar ambos resultados y establecer una valoración personal.

El operador Between permite simplificar la instrucción.

4. Obtener el id de los críticos que asignaron un puntaje de menos de 5 estrellas y tienen una puntuación superior a cero (0).

select IdCritico from puntaje where Estrellas < 5 and Puntos > 0;

VI. Operador lógico NOT. Operador LIKE.

5. Obtener las películas de la década del 60 (entre 1960 y 1969, incluidos) que **no** hayan sido realizadas en el Reino Unido.

select NombrePelicula from peliculas

where AnioPelicula between 1960 and 1969 and not PaisRealizacion = "UK";

6. Obtener la cantidad de actores que **no** han fallecido (computar actores y actrices juntamente).

select NombreActor, ApellidoActor from actores where FechaFallecActor is null;

7. Obtener los directores cuyos apellidos comienzan con la letra “N”.

select ApellidoDirector from directores where ApellidoDirector like "N%";

8. Obtener los directores cuyos apellidos contengan la letra “N”.

select ApellidoDirector from directores where ApellidoDirector like "%N%" ;

9. En la tabla reparto, obtener los nombres de personajes (rol) que no sean nombres compuestos.

select Rol from reparto where not Rol like "% %";

10. Obtener el id de los críticos que asignaron un puntaje menor a 5 estrellas y tienen una puntuación superior a cero (0) (utilizar el operador NOT).

select IdCritico from puntaje where not Estrellas > 5 and not Puntos < 0;

Unidad 4: Modelo físico. SQL

Práctica 4.

Te sugiero, escribir las sentencias en MySQL Workbench, comprobar el resultado conforme la consigna y, luego, pegar la instrucción debajo de cada ítem, según corresponda. Cualquier observación o aclaración que necesites hacer, podés añadirla al documento.

Resolver en MySQL.

VII. Modificación de datos y eliminación de datos.

Antes de avanzar en contenidos más complejos, vamos a introducir dos nuevas instrucciones: la sentencia UPDATE y la sentencia DELETE. Tal como su nombre en inglés lo indica, la instrucción UPDATE permite modificar datos en una tabla, mientras que DELETE, los elimina.

Te recuerdo que su ejecución es destructiva; esto quiere decir que los cambios aplicados tras ambas sentencias, **no se pueden deshacer**.

En este enlace, vas a encontrar el video tutorial de Khan Academy para ambas sentencias. Y, en este otro enlace, el desafío de codificación (ejercicio de aplicación).

Además, incluyo:

- UPDATE: En el enlace siguiente, vas a encontrar la sintaxis de esta sentencia y su aplicación en workbench (acceder).
- DELETE: En el enlace siguiente, vas a encontrar la sintaxis de esta sentencia y su aplicación en workbench (acceder)

****

1. Luego de leerlos, vas a ejecutar una consulta de selección sobre la tabla *actores*. Como podrás observar hay algunos actores cuya fecha de nacimiento se encuentra vacía. La actividad consiste en buscar las fechas de nacimientos de esos actores y agregarlas en la tabla para completar los campos.

update actores set FechaNacActor = '1979-04-12' where idactor = 112;

update actores set FechaNacActor = '1958-08-10' where idactor = 123 ;

2. Lo mismo vas a hacer pero con la tabla *películas*, ya que hay muchas Fechas de Lanzamiento en blanco.

update peliculas set FechaLanzamiento ="1999-09-03" where idPeliculas = 907;

update peliculas set FechaLanzamiento ="1999-09-15" where idPeliculas = 914;

update peliculas set FechaLanzamiento ="2006-11-30" where idPeliculas = 920;

3. Del mismo modo, vas a eliminar TODOS los datos de una tabla y, a continuación, los vas a recargar mediante la instrucción LOAD DATA. En el enlace, te presento un ejemplo que deberás adaptar a la tabla y conjunto de datos que elijas (el archivo cvs puede generarse en un bloc de notas).

delete from reparto;

load data infile 'D:/sql/reparto.csv'

into table reparto

fields terminated by ';'

lines terminated by '\r\n';

a. **RESPONDER**: ¿cuándo utilizarías esta instrucción? **Argumentar tu respuesta.**

Cuando se tiene una table vacía, sin datos. Si los datos están en un archivo de texto, la instrucción load data sirve para cargarlos a una table automáticamente sin tener que ingresar cada dato manualmente en su respectiva columna.

Unidad 4: Modelo físico. SQL

Práctica 4.

Te sugiero, escribir las sentencias en MySQL Workbench, comprobar el resultado conforme la consigna y, luego, pegar la instrucción debajo de cada ítem, según corresponda. Cualquier observación o aclaración que necesites hacer, podés añadirla al documento.

Resolver en MySQL.

En este enlace, vas a encontrar la teoría relacionada con subconsultas para Khan Academy; y aquí, el desafío de programación.

Además, te dejo este tutorial más sintético (acceder) que orienta con resultados en forma gráfica.

****

A continuación, los ejercicios...

VIII. Subconsultas

1. Escribir una consulta que devuelva todas las películas sin ranking.

select idPelicula from peliculas where idPelicula in (select idPelicula from puntaje where puntos = 0);

2. Escribir una consulta para hallar los nombres de todos los críticos que tienen puntaje nulo (estrellas = 0) . ¿Clausula DISTINCT?

select IdCritico, NombreCritico from criticos where IdCritico in (select idcritico from puntaje where estrellas = 0.00);

3. Escribir una consulta que devuelva los años en que se produjeron más de una película y que hayan recibido un puntaje mayor a 3 estrellas. Mostrar los resultados en orden creciente.

select aniopelicula from peliculas where (select count(AnioPelicula) from peliculas) >0 and IdPelicula in (select idpelicula from puntaje where Estrellas >3)

---

order by aniopelicula;

4. Obtener el nombre de todas las películas dirigidas por James Cameron.

select NombrePelicula from peliculas where idPelicula in (Select idPelicula from direccion where idDirector = 215);

5. Escribir una consulta que devuelva Id y nombre de todos los críticos que hayan rankeado con 7 ó más estrellas, devolver sólo los campos que contengan nombre.

select IdCritico, NombreCritico from criticos where not NombreCritico = "" and IdCritico in (select idCritico from puntaje where Estrellas >= 7);

6. Escribir una consulta que devuelva los nombres de aquellas películas que ***no*** tienen puntuación.

select NombrePelicula from peliculas where idPelicula in (select IdPelicula from puntaje where Puntos = 0);

7. Escribir una consulta que lista los actores y actrices que participan en 'Annie Hall'.

select NombreActor, ApellidoActor from actores where idactor in (select IdActor from reparto where IdPelicula in (select idPelicula from peliculas where NombrePelicula = "Annie Hall"));

Práctica 4.

Te sugiero, escribir las sentencias en MySQL Workbench, comprobar el resultado conforme la consigna y, luego, pegar la instrucción debajo de cada ítem, según corresponda. Cualquier observación o aclaración que necesites hacer, podés añadirla al documento.

IMPORTANTE

Leer con detenimiento los siguientes tutoriales:

- En primer lugar, les presento un enlace con la introducción a este contenido: JOIN, LEFT JOIN y RIGHT JOIN (acceder).
- En segundo lugar, comparto otro texto que explica la utilización de OUTER JOIN (acceder) y CROSS JOIN (acceder)
- Cuando finalices con esas lecciones, **podés comenzar con la ejercitación**:

XI. Consultas multitablas. Sentencia JOIN.

1. Escribir una consulta que retorne el ID y el nombre de los críticos que otorgaron 0 (cero) estrellas a cualquier película .
    a. Copiar y ejecutar esta sentencia y explicar qué ocurre:
    ```
    SELECT * FROM puntaje inner join criticos;
    ```
    Muestra todos los registros de las dos tablas.

b. Corregir la sentencia anterior para obtener el resultado buscado.

select puntaje.Estrellas, criticos.IdCritico, criticos.NombreCritico from puntaje inner join criticos on criticos.IdCritico=puntaje.IdCritico where Estrellas = 0;

c. Ejecutar la consulta con la instrucción CROSS JOIN, en lugar de INNER JOIN.

d. Comparar los resultados obtenidos en cada caso y elaborar una conclusión.

Cross join muestra todas las columnas de las tablas mientras que inner join sólo muestra algunas de las columnas.

2. Listar el nombre de todas las películas con su fecha de realización y género a que pertenecen.

select peliculas.NombrePelicula, peliculas.FechaLanzamiento, genero.genero from ((peliculas inner join generospeliculas on peliculas.idPelicula = generospeliculas.IdPelicula) inner join genero on genero.IdGenero = generospeliculas.IdGenero);

3. Escribir una consulta que retorne el ID y el nombre, las estrellas y el puntaje de <u>todos</u> los críticos, ordenados por IdCritico. **Argumentar tu resolución.**

select criticos.IdCritico, criticos.NombreCritico, puntaje.Estrellas, puntaje.Puntos from criticos cross join puntaje on criticos.IdCritico=puntaje.IdCritico order by IdCritico;

Se utiliza cross join en la instrucción para que SQL muestre todos los registros de las tablas.

4. Obtener los actores que **no** tengan roles asignados.

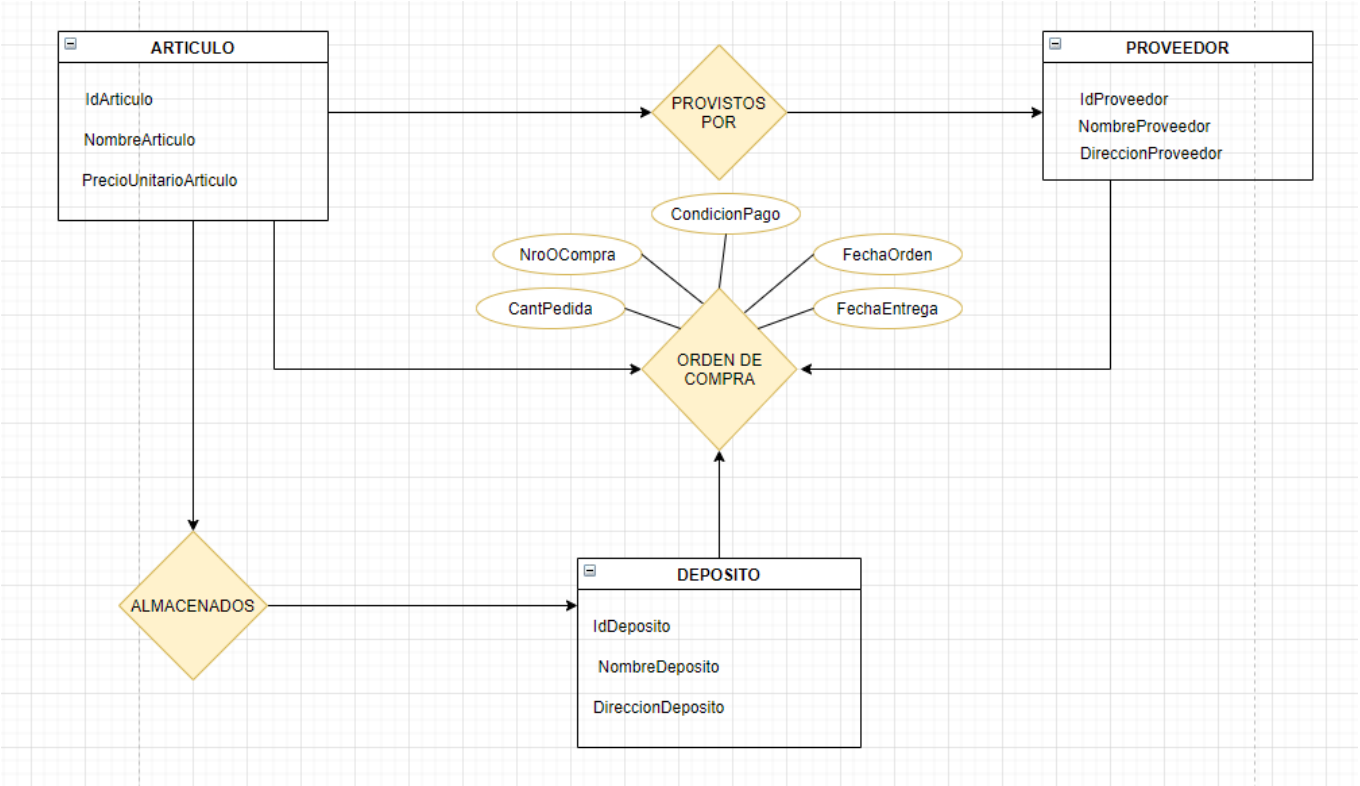
select actores.idactor, actores.NombreActor, actores.ApellidoActor from actores left join reparto on actores.idactor= reparto.IdActor where Rol is null;

Trabajo práctico de integración. Promoción Base de Datos I

Te presento tres narrativas de posibles sistemas de información a modelar. Leelas con detenimiento y elegí una de ellas.

A continuación:

a) Para la narrativa elegida, diseñar el **Modelo de Entidad Relación** (MER).



b) Realizar las vinculaciones funcionales; normalizar PASO a PASO y construir el **Mapa Canónico**.

ARTICULOS
IdArticulo
DescripcionArticulo
PrecioUnitario

PROVEEDORES
IdProveedor
NombreProveedor
DireccionProveedor

DEPOSITOS
IdDeposito
NombreDeposito
DireccionDeposito

DEPOSITO/ARTICULO
IdDeposito
IdArticulo
CantExistArt

ORDENES DE COMPRAS
IdOrdenCompra
IdArticulo
IdProveedor
IdDeposito
FechaOrdenCompra
FechaEntregaOrdenCompra
CondicionPago
CantidadPedida

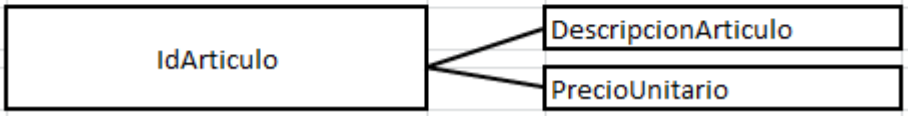
ARTICULOS

IdArticulo

DescripcionArticulo

PrecioUnitario

3ra FORMA NORMAL



PROVEEDORES

IdProveedor

NombreProveedor

DireccionProveedor

3ra FORMA NORMAL

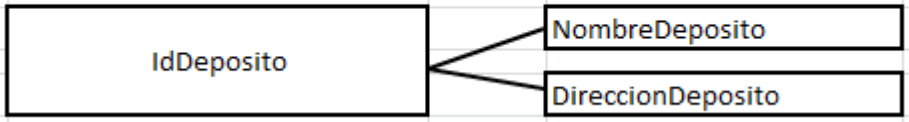
DEPOSITOS

IdDeposito

NombreDeposito

DireccionDeposito

3ra FORMA NORMAL



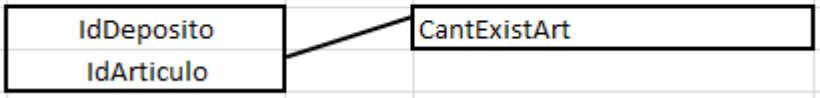
DEPOSITO/ARTICULO

IdDeposito

IdArticulo

CantExistArt

3ra FORMA NORMAL



ORDENES DE COMPRA

IdOrdenCompra

IdArticulo

IdProveedor

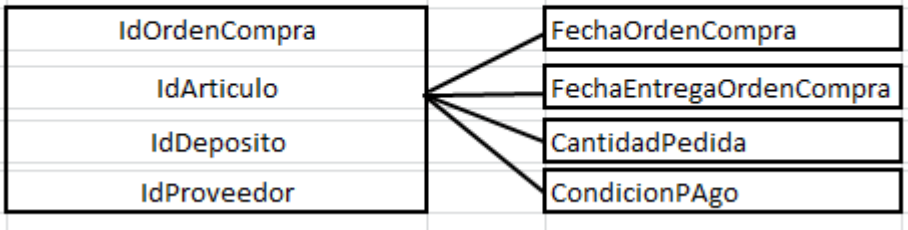
IdDeposito

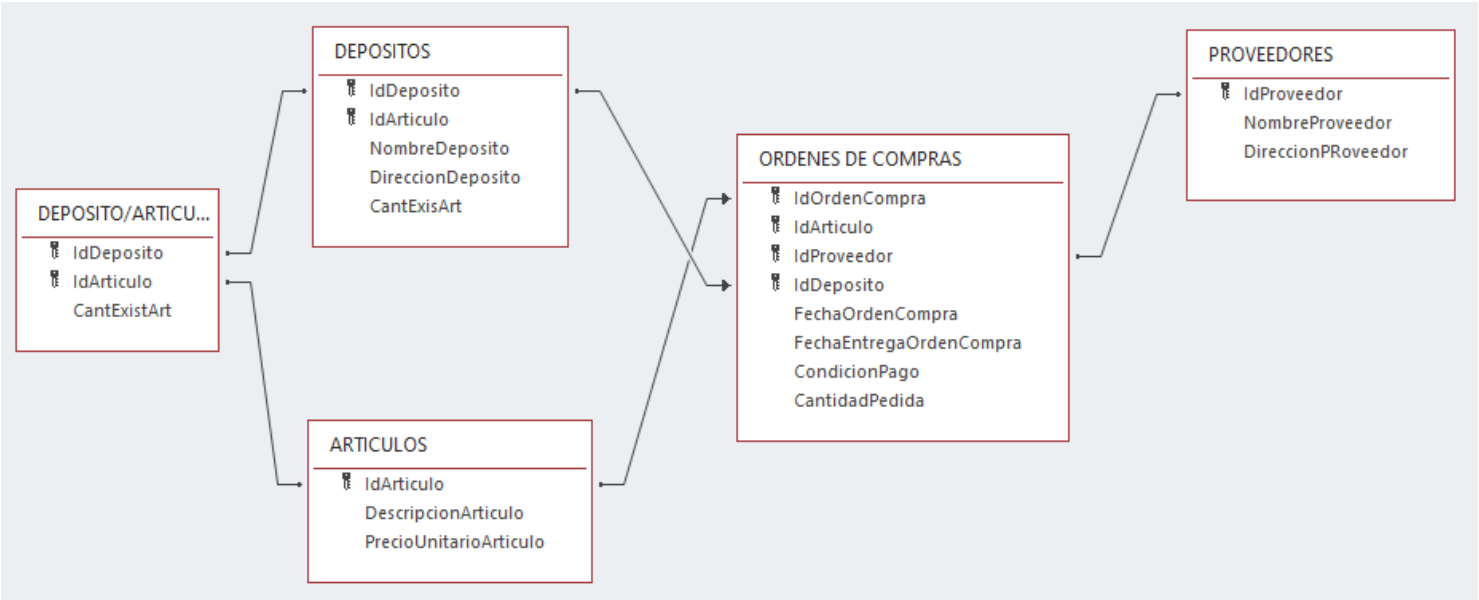
FechaOrdenCompra

FechaEntregaOrdenCompra

CantidadPedida

3ra FORMA NORMAL





c) Traducir al modelo físico en MySQL. Cargar datos de prueba consistentes (es decir, no se considerarán datos del tipo ”asadsfds”, “1234”, “pppp”).
d) Diseñar **al menos cuatro consultas de selección** que incluyan uniones de tablas, datos agrupados, funciones de agregado, filtros y parámetros. Cualquier otro componente (vistas, procedimientos almacenados, disparadores, otros) agregado a éstos obligatorios será tomado en consideración.
e) El trabajo es individual y opcional; si está correctamente elaborado, será defendido en un coloquio de promoción, con fecha a establecer, según calendario académico.

MODELO 2

La empresa VENTEX cuenta con varios depósitos para almacenar los artículos que se compran a sus proveedores mediante la orden de compra.

VISIONES DE CONTEXTO.
1- Los artículos están codificados, y tienen una descripción asociada a un precio unitario.
2- Los artículos pueden ser provistos por más de un proveedor.
3- Cada Orden de Compra va dirigida a un único proveedor y satisface las necesidades de un depósito determinado.
4- Cada artículo en una misma orden de compra puede tener una fecha de entrega diferente.
5- Los artículos pueden almacenarse en más de un depósito.

VISIONES DE USUARIO.
1 -ORDEN DE COMPRA NRO.....

FECHA DE O.C.: dd/mm/aa.
NOMBRE PROVEEDOR:.....
DOMICILIO PROVEEDOR:.....

DIRECCIÓN DE ENTREGA:.....
CONDICIÓN DE PAGO:.....

COD. ARTI.	DESCRIPCIÓN	CANT.	FECHA ENT.	PRE. UNI.	TOTAL
XXXXXXX	XXXXXXXXXXXXX	XXXXX	DD/MM/AA	XXXXXX	XXXXXXX
XXXXXXX	XXXXXXXXXXXXX	XXXXXX	DD/MM/AA	XXXXXX	XXXXXXX
XXXXXXX	XXXXXXXXXXXXX	XXXXXX	DD/MM/AA	XXXXXX	XXXXXXX
XXXXXXX	XXXXXXXXXXXXX	XXXXXX	DD/MM/AA	XXXXXX	XXXXXXX
		TOTAL O.COMPRA			**XXXXXXXX**

2- Por cada depósito se quiere conocer su nombre, dirección, artículos que almacena y cantidad de cada uno de ellos.

ORDENES-COMPRAS

NRO.O.COMPRA
FECHA-ORDEN
#ARTÍCULO
CANT-PEDIDA
FECHA-ENTREGA
#PROVEEDOR
NOMB-PROVEEDOR
DIREC-PROVEEDOR
#DEPOSITO
DIREC-DEPOSITO
CANT-EXIST-ART

ARTÍCULOS

#ARTÍCULO
DESC-ARTÍCULO
PRECIO-UNITARIO

c)

Creación de tablas

```
create table ordenescompra (
IdOrdenCompra INT NOT NULL primary key,
IdArticulo INT NOT NULL,
IdProveedor INT NOT NULL,
IdDeposito INT NOT NULL,
FechaOrdenCompra DATE,
FechaEntrega DATE,
CondicionPago VARCHAR(45),
CantidadPedida INT NOT NULL);

create table articulos (
IdArticulo INT NOT NULL PRIMARY KEY,
NombreArticulo VARCHAR (45),
PrecioUnitario INT NOT NULL);

create table depositos (
IdDeposito INT NOT NULL PRIMARY KEY,
DireccionDeposito VARCHAR(45),
```

```
NombreDeposito VARCHAR (45);

create table proveedores (
IdProveedor INT NOT NULL PRIMARY KEY,
NombreProveedor VARCHAR(45),
DireccionProveedor VARCHAR(45));

create table depositosarticulos (
IdDeposito INT NOT NULL PRIMARY KEY,
IdArticulo INT NOT NULL,
CantExistArt INT NOT NULL);
```

Inserción de datos

```
insert into articulos (IdArticulo, NombreArticulo, PrecioUnitario)
VALUES (100, "Leche", 50),
(101, "Gaseosa Cola", 110),
(102, "Agua Mineral", 60),
(103, "Pan Lactal", 60),
(104, "Queso Untable", 45),
(105, "Arroz", 40),
(106, "Fideos Tirabuzon", 40),
(107, "Yerba Mate", 80),
(108, "Dulce De Leche", 60),
(109, "Vino Malbec", 180),
(110, "Aceite de Girasol", 90);

insert into depositos (IdDeposito, DireccionDeposito, NombreDeposito)
VALUES (1000, "Quintana 600", “Deposito A”),
(1001, "Chapuis 1500", “Deposito B”),
(1002, "Guemes 450", “Deposito C”),
(1003, "Juan B. Justo 850" , “Deposito D”);

insert into proveedores (IdProveedor, NombreProveedor, DireccionProveedor)
VALUES (500, "Rodríguez", "Chile 550"),
(501, "Pérez", "Eva Perón 800"),
(502, "Fernández", "Ovidio Lagos 600"),
(503, "Sánchez", "Santa Fe 1350");

insert into depositosarticulos (IdDeposito, IdArticulo, CantExistArt)
VALUES (1002, 107, 20),
(1003, 102, 15),
(1000, 109, 50),
(1001, 101, 35);
(1002, 106, 55),
(1000, 108, 25);
```

```
insert into ordenescompra (IdOrdenCompra, IdArticulo, IdProveedor, IdDeposito,
FechaOrdenCompra, FechaEntrega, CondicionPago, CantidadPedida)
VALUES (200, 102, 500, 1002, '2021-05-03', '2021-05-09', "Contado", 15);
(201, 104, 501, 1002, '2021-05-15', '2021-05-17', "Cheque", 25),
(202, 110, 503, 1000, '2021-05-21', '2021-05-25', "Cta Cte", 50),
(203, 105, 500, 1001, '2021-06-10', '2021-06-20', "Contado", 35),
(204, 102, 501, 1003, '2021-06-15','2021-06-21', "Cheque", 60),
(205, 108, 503, 1001, '2021-06-30', '2021-07-05', "Cta Cte", 55),
(206, 103, 500, 1000, '2021-07-02', '2021-07-12', "Contado", 70),
(207, 106, 502, 1002, '2021-07-15', '2021-07-21', "Cheque", 35),
(208, 105, 500, 1003, '2021-07-22', '2021-07-29', "Contado", 40),
(209, 107, 502, 1001, '2021-08-01', '2021-08-07', "Cta Cte", 50),
(210, 104, 501, 1000, '2021-08-10', '2021-08-16', "Cheque", 20),
(211, 110, 500, 1003, '2021-08-19', '2021-08-28', "Contado", 60),
(212, 109, 503, 1002, '2021-08-29', '2021-09-04', "Cheque", 45);
```

d)

CONSULTA 1

Por cada depósito se quiere conocer su nombre, dirección, artículos que almacena y cantidad de cada uno de ellos.

```
select depositos.NombreDeposito, depositos.DireccionDeposito,
depositosarticulos.IdArticulo, depositosarticulos.CantExistArt from depositos inner join
depositosarticulos on depositos.IdDeposito = depositosarticulos.IdDeposito;
```

CONSULTA 2

Visualiza el ID y nombre del artículo del que mayor cantidad se pidió durante el mes de agosto.

```
select ordenescompra.IdArticulo, ordenescompra.CantidadPedida, articulos.NombreArticulo
from ordenescompra inner join articulos on ordenescompra.IdArticulo = articulos.IdArticulo
where  (ordenescompra.CantidadPedida = (select max(CantidadPedida)from
ordenescompra where (ordenescompra.FechaOrdenCompra between '2021-08-01' and
'2021-08-31'))) ;
```

CONSULTA 3

Muestra el ID, la cantidad pedida y el nombre de los proveedores a los que les pagamos al contado ordenado cronológicamente.

```
select ordenescompra.IdOrdenCompra, ordenescompra.CantidadPedida,
ordenescompra.CondicionPago, proveedores.NombreProveedor,
ordenescompra.FechaOrdenCompra from ordenescompra inner join proveedores on
ordenescompra.IdProveedor = proveedores.IdProveedor where
ordenescompra.CantidadPedida between 50 and 100 and ordenescompra.CondicionPago
like "Contado" order by FechaOrdenCompra asc;
```

CONSULTA 4
Visualiza el ID de la orden de compra, la cantidad pedida y el nombre del proveedor y los ordena de menor a mayor de acuerdo a la cantidad pedida.

```
select ordenescompra.IdOrdenCompra, ordenescompra.CantidadPedida, articulos.NombreArticulo, proveedores.NombreProveedor from ((proveedores inner join ordenescompra on ordenescompra.IdProveedor = proveedores.IdProveedor) inner join articulos on ordenescompra.IdArticulo = articulos.IdArticulo) where ordenescompra.FechaOrdenCompra between '2021-07-01' and '2021-08-31' order by ordenescompra.CantidadPedida asc;
```

CONSULTA 5
Visualiza el nombre de artículo junto al ID y el nombre del proveedor cuando este haya tardado más de 7 días en entregar los productos. Filtrado por trimestre.

```
select ordenescompra.IdProveedor, proveedores.NombreProveedor, articulos.NombreArticulo from ((ordenescompra inner join proveedores on ordenescompra.IdProveedor = proveedor.IdProveedor)  inner join articulos on ordendecompra.IdArticulo = articulos.IdArticulo)  where ordenescompra.FechaOrdenCompra between '2021-06-01' AND '2021-09-30' and timestampdiff(day,FechaOrdenCompra,FechaEntrega) >= 7;
```