

2024 segunda fecha

Modelo entidad relacion

Una vinoteca quiere modelar una base de datos para almacenar los productos que ofrece y las muestras a realizar sobre estos productos.

De cada vino que la vinoteca ofrece se sabe su nombre, bodega y los distintos varietales (tipos de uva, por ejemplo Malbec, Merlot, Moscatel y Tempranillo entre otras) que componen el vino, además del porcentaje de dicho varietal que lo compone. Por ejemplo, un vino puede estar compuesto por un 70% Malbec y 30% Merlot. Si bien cada vino puede tener varios varietales, cada varietal puede estar en un único vino o varios vinos.

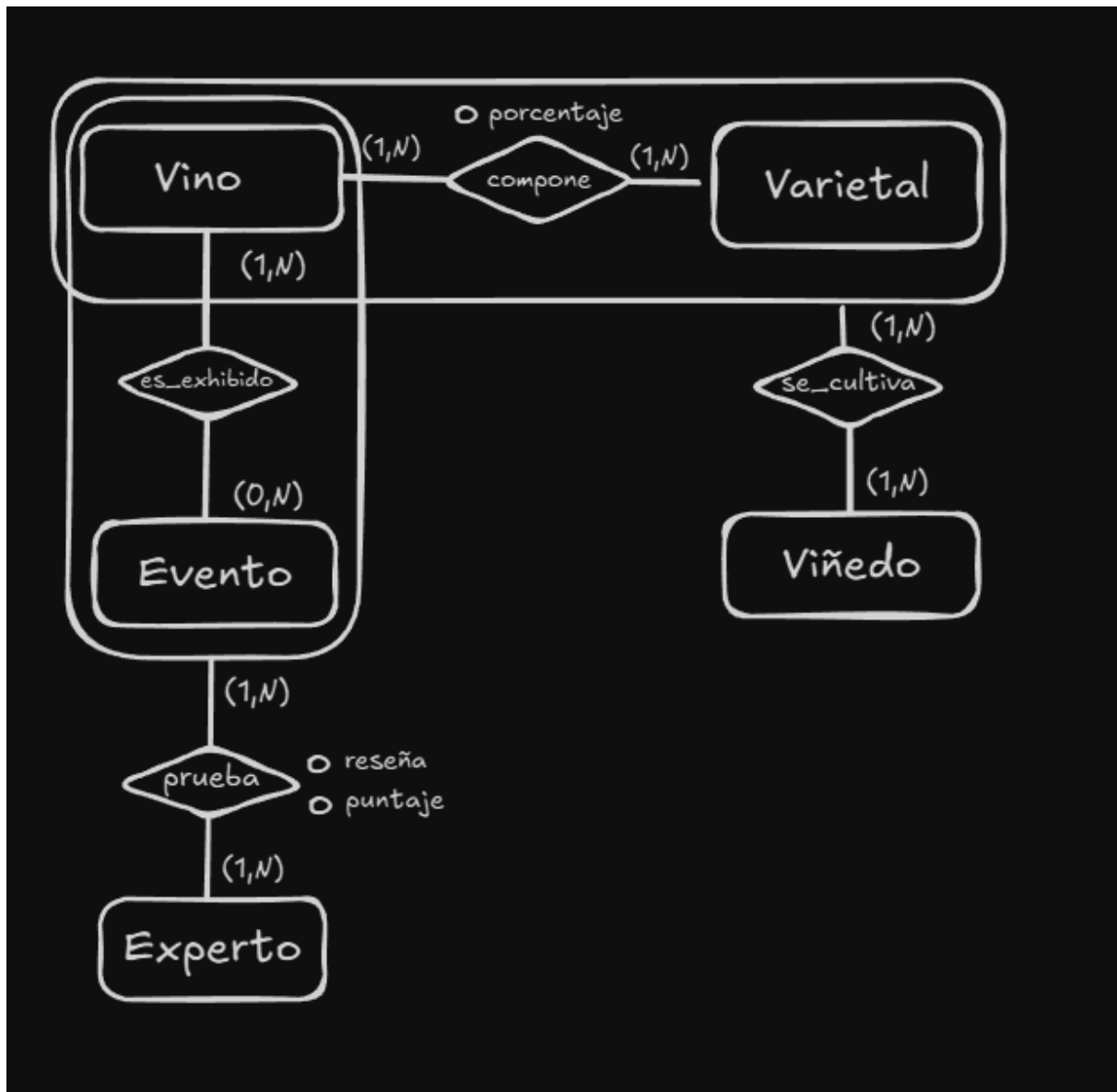
Adicionalmente, los diferentes varietales se cultivan en viñedos. Un viñedo puede tener diferentes varietales en sus tierras así como un varietal puede ser objeto de cultivo de diversos viñedos. A partir de lo anterior, se debe saber de cuál de los viñedos fue extraído el varietal que compone cada vino.

De cada varietal se sabe su nombre y su tipo. De los viñedos se sabe su nombre, dirección y altura.

La vinoteca organiza diferentes eventos para que clientes o expertos en vino puedan probar sus diferentes vinos. De cada evento se registra su nombre, la fecha en la que se realiza y los vinos que se eligen para ser presentados en ese evento. Un mismo vino puede ser presentado en diferentes eventos.

De cada experto que haya probado un vino presentado en un evento se registra una reseña y el puntaje que éste le otorga al vino (un valor de 0 a 100). De cada experto se sabe su nombre, apellido, cuil, fecha de nacimiento, nacionalidad y experiencia.

Es importante para la vinoteca registrar la opinión de los expertos que han probado y reseñado al vino presentado, información que servirá luego para poder promocionar sus vinos.



Normalizacion

Normalización

Dado el siguiente esquema que representa la actividad en un parque de diversiones:

PARQUEDIVERSIONES (id_atractivo, nombre_atractivo, descripción_atractivo, id_categoria_atractivo, nombre_categoria, cuil_empleado, nombre_apellido_empleado, nro_legajo_empleado, dia_semana, id_agente_mantenimiento)

- De cada atractivo (identificado por id_atractivo), se conoce su nombre (nombre_atractivo), una descripción (descripción_atractivo) y las categorías a la que pertenece. Uno o más atractivos pueden tener el mismo nombre y/o descripción. Puede haber diferentes atractivos por categoría.
- El id de la categoría es un valor único (id_categoria_atractivo). De cada categoría se conoce su nombre. Un nombre de categoría puede repetirse para diferentes id de categoría.
- De cada empleado que trabaja en el parque de diversiones, se conocen además de sus datos personales (cuil_empleado, nombre_apellido_empleado), un número de legajo (nro_legajo_empleado) el cual, al igual que el cuil, es único en el sistema.
- En cada atractivo hay asignados diversos empleados, pero dado un atractivo y un empleado asignado se conoce el día de la semana que le corresponde.
- Un atractivo puede ser mantenido por distintos empleados en un día determinado.
- Un empleado puede tener asignado más de un atractivo por día.
- El parque de diversiones también dispone de personal de mantenimiento, que están disponibles para atender a los diversos atractivos del parque.

Aplicar y explicar el proceso de normalización visto en la materia. Considerar que el esquema ya se encuentra en 1FN.

▼ Normalización

Hallar dependencias funcionales

- df1: id_atractivo → nombre_atractivo, descripción_atractivo
- df2: id_categoria_atractivo → nombre_categoria
- df3: cuil_empleado → nombre_apellido_empleado, nro_legajo_empleado
- df4: nro_legajo_empleado → nombre_apellido_empleado, cuil_empleado
- df5: id_atractivo, cuil_empleado → dia_semana
- df6: id_atractivo, nro_legajo_empleado → dia_semana

CC1: {id_atractivo, id_categoria_atractivo, cuil_empleado, id_agente_mantenimiento}

CC2: {id_atractivo, id_categoria_atractivo, nro_legajo_empleado, id_agente_mantenimiento}

Vemos que PARQUEDIVERSIONES no está en BCNF ya que existe al menos una df, como por ejemplo df2, cuyo determinante {id_categoria_atractivo} no es superclave del esquema. Particionamos PARQUEDIVERSIONES a partir de df2.

- P1(id_categoria_atractivo, nombre_categoria)
- P2(id_atractivo, nombre_atractivo, descripcion_atractivo, id_categoria_atractivo, cuil_empleado, nombre_apellido_empleado, nro_legajo_empleado, dia_semana, id_agente_mantenimiento)

No se pierde informacion ya que P1 n P2 es {id_categoria_atractivo} clave de P1

Por validacion simple en P1 vale df2 y en P2 valen df1, df3, df4, df5 y df6

P1 esta en BCNF ya que solo vale df2 en el esquema cuyo determinante {id_categoria_atractivo} es superclave del esquema

Vemos que P2 no esta en BCNF ya que existe al menos una df, como por ejemplo df1, cuyo determinante {id_atractivo} no es superclave del esquema. Particionamos P2 a partir de df1.

- P3(id_atractivo, nombre_atractivo, descripcion_atractivo)
- P4(id_atractivo, id_categoria_atractivo, cuil_empleado, nombre_apellido_empleado, nro_legajo_empleado, dia_semana, id_agente_mantenimiento)

No se pierde informacion ya que P3 n P4 es {id_atractivo} clave de P3

Por validacion simple en P3 vale df1 y en P4 valen df3, df4, df5 y df6

P3 esta en BCNF ya que solo vale df1 en el esquema cuyo determinante {id_atractivo} es superclave del esquema

Vemos que P4 no esta en BCNF ya que existe al menos una df, como por ejemplo df5, cuyo determinante {id_atractivo, cuil_empleado} no es superclave del esquema. Particionamos P4 a partir de df5.

- P5(id_atractivo, cuil_empleado, dia_semana)
- P6(id_atractivo, id_categoria_atractivo, cuil_empleado, nombre_apellido_empleado, nro_legajo_empleado, id_agente_mantenimiento)

No se pierde informacion ya que P5 n P6 es {id_atractivo, cuil_empleado} clave de P5

Por validacion simple en P5 vale df5 y en P4 valen df3, df4

¿Que paso con df6?

No se pierde la df6 ya que mediante la df5: id_atractivo, cuil_empleado → dia_semana, que vale en P5, recuperamos el dia_semana. Por lo tanto tenemos dfs equivalentes ya que con distintos determinantes, producimos el mismo resultado.

VALEN

No se pierde la df6 ya que al ser equivalente con df5 obtenemos dia_semana.

CAPE

La DF6 no se pierde, ya que es posible acceder con su determinante a todos sus determinados, ya sea directa o indirectamente. dia_semana no esta en P6, sin embargo la dependencia funcional no se pierde ya que existe una relacion indirecta a traves de la dependencia funcional

id_atractivo, nro_legajo_empleado → cuil_empleado

y

id_atractivo, cuil_empleado → dia_semana

La df6 no quedo valida pero no se pierde.

P5 esta en BCNF ya que solo vale df5 en el esquema cuyo determinante {id_atractivo, cuil_empleado} es superclave del esquema

Vemos que P6 no esta en BCNF ya que existe al menos una df, como por ejemplo df3, cuyo determinante {cuil_empleado} no es superclave del esquema. Particionamos P6 a partir de df3.

- P7(cuil_empleado, nombre_apellido_empleado, nro_legajo_empleado)
- P8(id_atractivo, id_categoria_atractivo, cuil_empleado, id_agente_mantenimiento)

No se pierde informacion ya que P7 n P8 es {cuil_empleado} clave de P7

Por validacion simple en P7 valen df3 y df4.

P7 esta en BCNF ya que vale df3 cuyo determinante {cuil_empleado} es superclave del esquema. Tambien vale df4 cuyo determinante

{nro_legajo_empleado} es superclave del esquema

En P8 quedaron todos los atributos de la clave candidata. Cualquier dependencia funcional que valga es trivial. Por lo tanto P8 esta en BCNF

Esquemas en BCNF

- P1(id_categoria_atractivo, nombre_categoria)
- P3(id_atractivo, nombre_atractivo, descripcion_atractivo)
- P5(id_atractivo, cuil_empleado, dia_semana)
- P7(cuil_empleado, nombre_apellido_empleado, nro_legajo_empleado)
- P8(id_atractivo, id_categoria_atractivo, cuil_empleado, id_agente_mantenimiento)

CP: {id_atractivo, id_categoria_atractivo, cuil_empleado, id_agente_mantenimiento}

Hallar dependencias multivaluadas

Algebra relacional

Álgebra Relacional

Algoritmo para analizar la pérdida de dfs. Res = x Mientras Res cambia Para i= 1 to cant_de_ particiones_realizadas Res = Res U ((Res ∩ Ri)* ∩ Ri).	Algoritmo para encontrar X* Result:= X While (hay cambios en result) do For (cada dependencia funcional Y->Z en F) do if (Y ⊆ result) then result := result U Z
------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Dado el siguiente esquema:

PRODUCTO (#producto, #categoria, descripcion_producto)
CATEGORIA (#categoria, descripcion_categoria)
MAYORISTA (#mayorista, cuil, razón_social)
PRECIO_MAYORISTA (#producto, #mayorista, precio)
VENTA (#venta, #producto, #mayorista, #cliente, cantidad, fecha)
CLIENTE (#cliente, cuil, nombre_cliente, apellido_cliente)

A) Listar el id de los mayoristas que han vendido productos de todas las categorias de las cuales ofrecen productos.

```
CATEGORIAS_OFRECIDAS ← h #mayorista, #categoria (PRECIO_MAYORISTA |X| PRODUCTO)
```

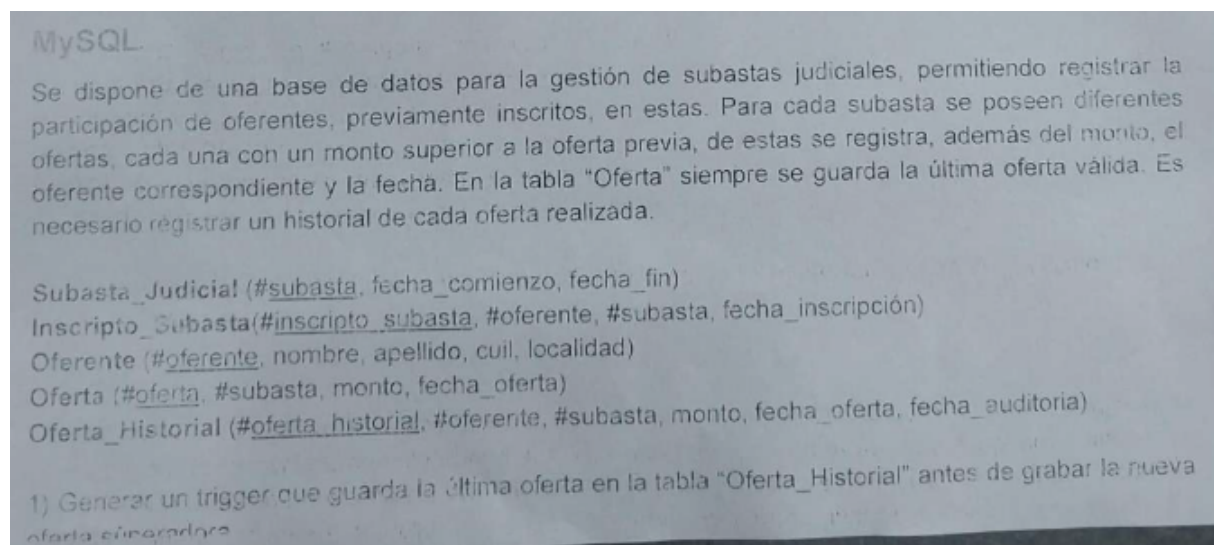
```
CATEGORIAS_VENDIDAS ← h #mayorista, #categoria (VENTA |X| PRODUCTO)
```

```
NO_CUMPLEN ← h #mayorista (CATEGORIAS_OFRECIDAS - CATEGORIAS_VENDIDAS)
```

```
RESULTADO ← h #mayorista MAYORISTA - NO_CUMPLEN
```

SQL

*Oferta tiene #oferente



```
DELIMITER //
```

```
CREATE TRIGGER nueva_oferta  
BEFORE UPDATE ON OFERTA  
FOR EACH ROW  
BEGIN
```

```
INSERT ON OFERTA_HISTORIAL(oferente, subasta, monto, fecha_ofert
```

```
a, fecha_auditoria)
    VALUES(old.oferente, old.subasta, old.monto, old.fecha_oferta, NOW
())

END;
DELIMITER ;
```