Examen de
Fundamentos de Bases de Datos

Julio 2024

Indicaciones Generales:

- La duración del examen es de **tres (3)** horas.
- En la prueba **NO** se permite consultar material alguno.
- Empezar cada ejercicio en una hoja nueva.
- Escribir con lápiz y de un solo lado de las hojas.
- Numerar todas las hojas. Incluir en cada hoja la cédula y el nombre. En la primer hoja, incluir la cantidad de hojas que se entregan.

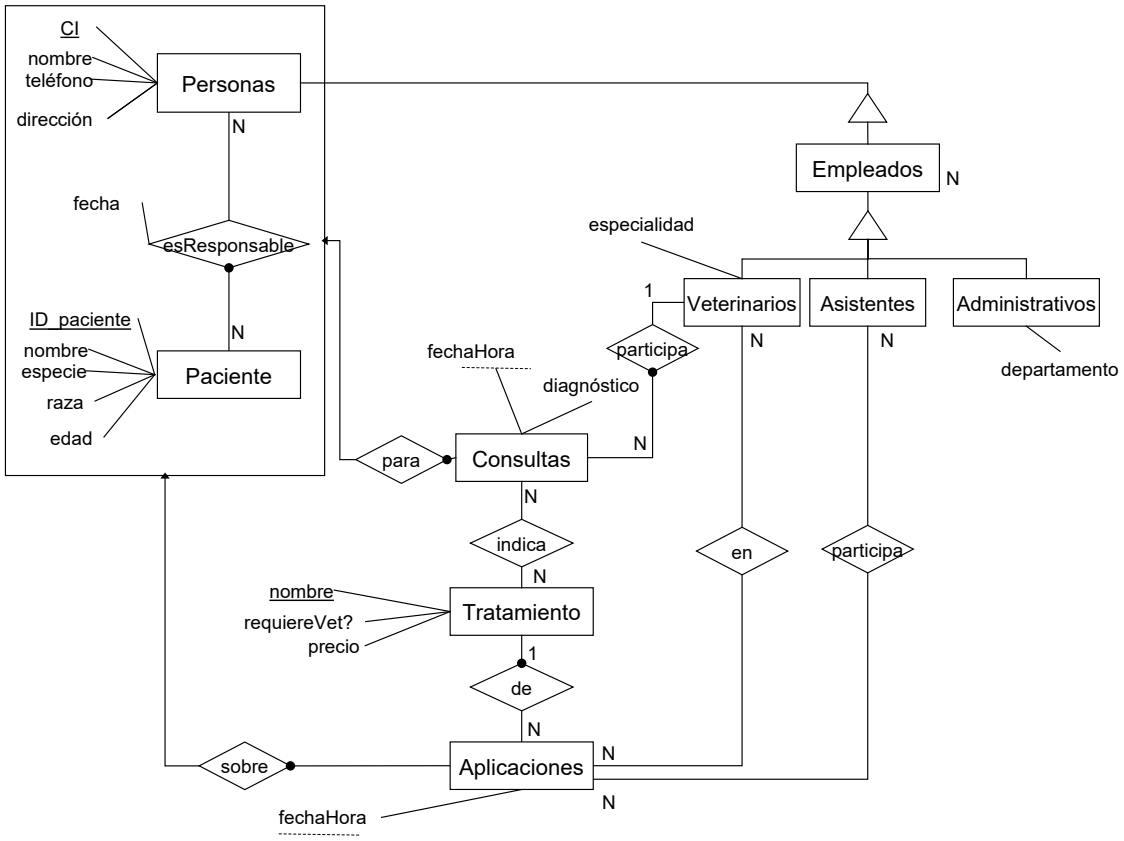
Ejercicio 1. (25 puntos)

Una clínica veterinaria le solicita a Ud. que diseñe una base de datos para llevar el registro de sus pacientes, tratamientos, empleados y proveedores. Lo que sigue es el resultado del relevamiento realizado por los analistas de la clínica.

La clínica tiene un conjunto de pacientes, de los cuales se conoce su número de identificación (ID_paciente), nombre, especie (perro, gato, etc.), raza y edad. Para cada paciente se registra una o más personas responsables, y la fecha desde la cual esa persona es responsable. De cada responsable se conoce su CI, nombre y teléfono. La clínica emplea diferentes tipos de empleados. De cada empleado se conoce su cédula de identidad, nombre y teléfono. Los empleados pueden ser de tres tipos: veterinarios, asistentes y administrativos. Algunos empleados de la clínica usan los servicios con sus mascotas, y por lo tanto se registran como responsables de éstas. De los veterinarios se conoce su especialidad, y de los administrativos se conoce la sección de la clínica donde trabajan (recepción, contabilidad, etc.). En la clínica se realizan diferentes tratamientos, que incluyen desde vacunaciones y desparasitaciones hasta cirugías y cuidados específicos según la salud de la mascota. Los tratamientos se identifican por su nombre y se conoce su precio. Para su aplicación, algunos tratamientos requieren de la presencia de un veterinario mientras que otros pueden ser aplicados por los asistentes (por ejemplo, baños); este dato se debe registrar. En todo caso, de cada aplicación de un tratamiento sobre un paciente interesa registrar la fecha y hora y los veterinarios y/o asistentes de la clínica que participan. Además, cada vez que se aplica un tratamiento se exige la presencia de uno de los responsables de la mascota, y también interesa registrar esto. Notar que un mismo paciente no puede estar a la vez (misma fecha y hora) en dos aplicaciones de tratamiento. Por lo tanto, una aplicación con determinada fecha, hora y paciente, es única. Además, los veterinarios de la clínica atienden consultas. Para cada consulta a la que asiste un paciente se registra la persona responsable que lo acompaña, la fecha y hora de la consulta, el veterinario que participa y el diagnóstico. En cada consulta se pueden recomendar tratamientos específicos e interesa registrar esto. Al igual que con las aplicaciones de tratamiento, un mismo paciente no puede estar a la vez en dos consultas.

(a) Elabore un Modelo Entidad-Relación completo del problema, incluyendo Restricciones No Estructurales.

Solución.



Restricciones no estructurales:

$Veterinarios \cap Asistentes \cap Administrativos = \emptyset$

Un paciente no pueden recibir distintas aplicaciones en la misma fechaHora

$(\forall ap \in Aplicaciones)((\forall p \in Personas)(\forall pa \in Paciente)$

$(< p, pa > \in esResponsable \wedge < p, pa >, ap > \in sobre \rightarrow$

$(\neg \exists ap1 \in Aplicaciones \wedge \exists p1 \in Personas \wedge < p1, pa >, ap1 > \in sobre \wedge ap.fechaHora = ap1.fechaHora)))$

Un paciente no puede estar en distintas consultas en la misma fechaHora

$(\forall ap \in Consultas)((\forall p \in Personas)(\forall pa \in Paciente)$

$(< p, pa > \in esResponsable \wedge < p, pa >, ap > \in para \rightarrow$

$(\neg \exists ap1 \in Consultas \wedge \exists p1 \in Personas \wedge < p1, pa >, ap1 > \in para \wedge ap.fechaHora = ap1.fechaHora)))$

Un asistente o un veterinario no pueden realizar distintas aplicaciones en la misma fechaHora

Un veterinario no puede participar en distintas consultas en la misma fechaHora

Ejercicio 2. (25 puntos)

En un centro comercial (shopping) se desea crear una base de datos relacional con los datos de interés, correspondientes a la siguiente realidad.

Se tiene un conjunto de locales, cada uno de ellos con un identificador, la zona donde se encuentra dentro del shopping, el tipo de local (común, isla, abierto, etc.). Por otro lado, el shopping tiene comercios, de los cuales interesa el identificador, el nombre, el nombre del encargado, el rubro. Cada comercio pue-

de estar en más de un local del shopping y en cada local puede haber un único comercio. Por otro lado, un comercio no puede tener más de un local en la misma zona. Además, interesan los datos sobre las promociones del shopping. De cada una de éstas se guarda un identificador, un nombre, la temporada y un porcentaje de descuento. Los comercios pueden adherir a varias promociones y esto es fijo, es decir que no cambia en el tiempo. El shopping tiene además los siguientes datos de sus empleados de marketing: cédula, nombre y teléfono. Cada año se designa para cada promoción a uno de estos empleados como encargado de supervisarla. La Tabla 1 detalla los nombres de atributos a utilizar.

idLocal	encargado	descuento
zona	rubro	ci
tipo	idProm	nomEmp
idCom	nomProm	tel
nomCom	temporada	año

Cuadro 1: Nombres de atributos.

Se pide, justificando las respuestas:

(a) Deducir qué dependencias funcionales se cumplen entre los atributos dados (conjunto F).

Solución.

```
F = { idLocal -> zona, tipo, idCom
zona, idCom -> idLocal
idCom -> nomCom, encargado, rubro
idProm -> nomProm, temporada, descuento
ci -> nomEmp, tel
idProm, año -> ci }
```

(b) Dar un cubrimiento minimal de F.

Solución.

Paso 1:

```
Fmin = {  idLocal -> zona
    idLocal -> tipo
    idLocal -> idCom
    zona, idCom -> idLocal
    idCom -> nomCom
    idCom -> encargado
    idCom -> rubro
    idProm -> nomProm
    idProm -> temporada
    idProm -> descuento
    ci -> nomEmp
    ci -> tel
    idProm, año -> ci }
```

Paso 2: No hay atributos redundantes a la izq. de ninguna df.

En zona, idCom $\rightarrow$ idLocal no puedo sacar idCom, porque zona no está a la izquierda de ninguna otra df, por lo tanto no puedo obtener idLocal partiendo solamente de zona. Calculo $(zona)_F^+ = \{zona\}$. No puedo sacar zona porque solamente con idCom no puedo obtener idLocal. Eso lo veo calculando $(idCom)_F^+ = \{idCom, nomCom, encargado, rubro\}$

En zona, idProm, año $\rightarrow$ ci no puedo sacar idProm, porque año no está a la izquierda de ninguna otra df, por lo tanto no puedo obtener ci partiendo solamente de año. Eso lo veo calculando $(anio)_F^+ = \{anio\}$

No puedo sacar año porque solamente con idProm no puedo obtener ci.
$(idProm)^+_F = \{idProm, nomProm, temporada, descuento\}$
Paso 3: No hay dfs redundantes porque en todas las dfs el atributo de la derecha sólo es determinado en esa df.

(c) Considerando la Relación Universal, que contiene a todos los atributos, hallar todas las claves según F.

Solución.
Para hallar las claves:
(idProm, año) pertenecen a todas las claves porque no están a la derecha de ninguna df.
(tipo, nomCom, encargado, rubro, nomProm, temporada, descuento, nomEmp, tel) no pertenecen a ninguna clave porque no están a la izquierda de ninguna df.
Probamos con idLocal, zona e idCom.
(idLocal, idProm, año)+ contiene a todos los atributos de R. (zona, idProm, año)+ no contiene a todos los atributos de R. (idCom, idProm, año)+ no contiene a todos los atributos de R. (zona, idCom, idProm, año)+ contiene a todos los atributos de R.
Claves únicas:
(idLocal, idProm, año)
(zona, idCom, idProm, año)

(d) Aplicar el algoritmo de 3NF con JSP y preservación de dfs visto en el curso.

Solución.
1er paso) Hallar Fmin. Tomo el de la parte 2)
R1 (idLocal, zona, tipo, idCom) R2 (zona, idCom, idLocal) R3 (idCom, nomCom, encargado, rubro) R4 (idProm, nomProm, temporada, descuento) R5 (ci, nomEmp, tel) R6 (idProm, año, ci)
2o paso) Agrego un sub-esquema que sea una clave.
R1 (idLocal, zona, tipo, idCom) R2 (zona, idCom, idLocal) R3 (idCom, nomCom, encargado, rubro) R4 (idProm, nomProm, temporada, descuento) R5 (ci, nomEmp, tel) R6 (idProm, año, ci) R7 (idLocal, idProm, año)
3er paso) Elimino R2 por estar incluido en R1
R1 (idLocal, zona, tipo, idCom) R2 (idCom, nomCom, encargado, rubro) R3 (idProm, nomProm, temporada, descuento) R4 (ci, nomEmp, tel) R5 (idProm, año, ci) R6 (idLocal, idProm, año)

(e) Llevar a BCNF, aplicando el algoritmo visto en el curso, el esquema relacional construido en la parte 4.

Solución.
Proyecto dfs y hallo claves en la descomposición anterior.
R1 (idLocal, zona, tipo, idCom) claves = (idLocal) y (zona, idCom)
FR1 = { idLocal $\rightarrow$ zona, tipo, idCom zona, idCom $\rightarrow$ idLocal }
R2 (idCom, nomCom, encargado, rubro)
FR2 = { idCom $\rightarrow$ nomCom, encargado, rubro }
R3 (idProm, nomProm, temporada, descuento)
FR3 = { idProm $\rightarrow$ nomProm, temporada, descuento }
R4 (ci, nomEmp, tel)
FR4 = { ci $\rightarrow$ nomEmp, tel }
R5 (idProm, año, ci)
FR5 = { idProm, año $\rightarrow$ ci }
R6 (idLocal, idProm, año)
FR6 = { }

Podemos observar que todas las relaciones anteriores están en BCNF, excepto R1.

Llevamos R1 a BCNF, aplicando el algoritmo:

Tomamos una df que viola BCNF: zona, idCom $\rightarrow$ idLocal

R11 (zona, idCom, idLocal) R12 (zona, tipo, idCom)

Finalmente la descomposición en BCNF es la siguiente:

R11 (zona, idCom, idLocal) R12 (zona, tipo, idCom) R2 (idCom, nomCom, encargado, rubro) R3 (idProm, nomProm, temporada, descuento) R4 (ci, nomEmp, tel) R5 (idProm, año, ci) R6 (idLocal, idProm, año)

(f) Para cada tabla obtenida en la descomposición de la parte 5, hallar las dependencias multivaluadas que se cumplen.

Solución.
En R6 se cumple la MVD: idLocal $\twoheadrightarrow$ idProm

Esto es porque un local tiene varias promociones asociadas y también varios años asociados, ya que si un comercio adhiere a una promoción sus locales también adhieren. Además, las promociones y los años son independientes, es decir que todas las promociones se corresponden con todos los años. Esto último lo sabemos por la frase de la letra: "Los comercios pueden adherir a varias promociones y esto es fijo, es decir que no cambia en el tiempo."

(g) Teniendo en cuenta la parte 6, llevar a 4NF el esquema relacional de la parte 5.

Solución.
Para llevar el esquema de la parte 5 a 4NF, debemos descomponer al esquema R6 en dos subesquemas.

R61 (idLocal, idProm)
R62 (idLocal, año)

Ejercicio 3. (30 puntos)

Una heladería almacena información sobre sus helados, los ingredientes que los componen y los proveedores a los que compran cada ingrediente en la siguiente base de datos:

Helados(idHelado, nombre)
Ingredientes(idIngrediente, nombre)
Proveedores(idProveedor, nombre,contacto)
Ingrediente_Proveedor(idIngrediente, idProveedor, fecha_ultimo_pedido)
Almacena información sobre que proveedores proveen cada ingrediente y la fecha en que se realizó el último pedido de cierto ingrediente a un proveedor.

Helado_Ingrediente(idHelado,idIngrediente,cantidad)
Almacena información sobre los ingredientes necesarios para elaborar cada helado, indicando la cantidad de cada ingrediente (cantidad siempre es > 0)

En la base de datos no hay tablas vacías y se cumplen las siguientes dependencias de inclusión:

$\Pi_{idHelado}$Helado_Ingrediente $\subseteq \Pi_{idHelado}$ Helados

$\Pi_{idIngrediente}$Helado_Ingrediente $\subseteq \Pi_{idIngrediente}$ Ingredientes

$\Pi_{idProveedor}$Ingrediente_Proveedor $\subseteq \Pi_{idProveedor}$Proveedores

$\Pi_{idIngrediente}$Ingrediente_Proveedor $\subseteq \Pi_{idIngrediente}$ Ingredientes

Se pide:

(a) Resuelva en SQL las siguientes consultas

1. Devolver los nombres de los helados tales que todos sus ingredientes son provistos por un único proveedor que es el mismo para todos los ingredientes.

Solución.

```
SELECT h.nombre
FROM    Helados h JOIN Helado_Ingrediente hi ON h.id_helado = hi.
    id_helado
                JOIN Ingrediente_Proveedor ip ON hi.id_ingrediente = ip.
                    id_ingrediente
GROUP BY h.id_helado, h.nombre
HAVING COUNT(DISTINCT ip.id_proveedor) = 1;

--otra opcion
SELECT DISTINCT h.nombre
FROM Helados h JOIN Helado_Ingrediente hi ON h.id_helado = hi.id_helado
        JOIN Ingrediente_Proveedor ip ON hi.d_ingrediente = ip.
            id_ingrediente
WHERE NOT EXISTS (
        SELECT 1
        FROM Ingrediente_Proveedor ip2 JOIN
                 Helado_Ingrediente hi2 ON ip2.id_ingrediente = hi2.
                     id_ingrediente
        WHERE hi2.id_helado = h.id_helado
                AND ip2.id_proveedor <> ip.id_proveedor);

-- y otra opcion
SELECT DISTINCT h.nombre
FROM Helados h NATURAL JOIN Helado_Ingrediente hi
WHERE hi.IdIngrediente  IN
 (
        SELECT ip1.IdIngrediente
        FROM Ingrediente_Proveedor ip1
        WHERE NOT EXISTS (
                SELECT 1
                FROM Ingrediente_Proveedor ip2
                WHERE ip1.idIngrediente = ip2.idIngrediente
                        AND ip1.idProveedor <> ip2.idProveedor)
)
```

II. Devolver los nombres de los ingredientes que se pidieron en la fecha más cercana.
Solución.

```
        SELECT i.nombre
        FROM Ingredientes i JOIN Ingrediente_Proveedor ip ON
                i.id_ingrediente = ip.id_ingrediente
        WHERE ip.fecha_ultimo_pedido =
                (SELECT MAX(fecha_ultimo_pedido)
                FROM Ingrediente_Proveedor);
```

(b) Resuelva en Algebra Relacional las siguientes consultas

I. Devolver los nombres de los helados de los que hay al menos un ingrediente que es provisto por más de un proveedor.
Solución.
$A = \rho_{idIingrediente,idProveedor \to ing,prov}(Ingrediente\_proveedor)$
$B = Ingrediente\_proveedor \bowtie_{idIngrediente=ing \wedge idProveedor \neq prov} A$
$Sol = \Pi_{nombre}(\Pi_{idIngrediente}(A) * Helados)$

II. Devolver los nombres de los proveedores que proveen todos los ingredientes para todos los helados que tienen ingredientes.

Solución.

$A = \Pi_{idProveedor,idHelado,idIngrediente}(Helado\_ingrediente * Ingrediente\_proveedor)$
$Sol = A \div \Pi_{idHelado,idIngrediente}(Helado\_ingrediente)$

(c) Resuelva en Cálculo Relacional la siguiente consulta

I. Devolver los nombres de los proveedores que proveen al menos un ingrediente para cada helado.

Solución.

$$\langle p.nombre \rangle / Proveedor(p) \wedge (\forall h)\ (Helados(h) \rightarrow (\exists hi)(helado\_ingrediente(hi) \wedge hi.idHelado = h.idHelado \wedge (\exists ip)(ingrediente\_proveedor(ip) \wedge hi.idIngrediente = ip.idIngrediente \wedge p.idProveedor = ip.idProveedor)))$$

Ejercicio 4. (20 puntos)

Dadas las siguientes transacciones:

$T_1 : w_1(x), r_1(y), w_1(y), r_1(x), c_1$
$T_2 : w_2(z), r_2(z), c_2$

(a) Justifique el motivo por el cual todas las posibles historias conformadas por T_1 y T_2 son serializables.

Solución.

T1 y T2 operan sobre diferentes elementos, por lo tanto, no hay operaciones en conflicto.

(b) Agregue operaciones a T_2, conformando una nueva transacción T_2' y escriba una historia H_1, conteniendo T_1 y T_2', de forma de que H1 no sea serializable. Justifique su respuesta.

Solución.

T2': w2(x), w2(z), r2(z), w2(y), c2

H1: w1(x),w2(x), w2(z), r2(z), w2(y), r1(y), w1(y), r1(x), c1,c2

Dos pares de operaciones en conflicto son w1(x),w2(x) y w2(y), r1(y). Estos pares de operaciones generan un arco de T1 a T2 y otro de T2 a T1, por lo tanto, H1 no es serializable.

(c) Modifique T_2' de forma de que aborte en lugar de confirmar. Escriba una historia H_2 que sea serializable pero que no evite abortos en cascada. Justifique su respuesta.

Solución.

T2': w2(x), w2(z), r2(z), w2(y), a2 H2: w2(x), w2(z), r2(z), w2(y), w1(x), r1(y), a2, w1(y), r1(x), c1

Los arcos van solo de T1 a T2, por lo tanto no hay ciclos y H es serializable. Por otra parte, r1(y) lee de T2 que aborta, por lo tanto, H no evita abortos en cascada.

(d) Agregue operaciones de bloqueo y desbloqueo a T_1 de manera de que siga el protocolo 2PL estricto pero no conservador. Justifique su respuesta.

Solución.

T1: wl1(x), w1(x), wl1(y), r1(y), w1(y), r1(x), c1, u1(x), u1(y)

Dado que los desbloqueos se dan luego del commit, respeta el protocolo 2PL estricto. Sin embargo, no todos los bloqueos se dan antes de la primera operación de la transacción, por lo tanto, no respeta 2PL conservador.