Ayudantía de Repaso I1

Álgebra Relacional

- Actores(id, nombre, edad)
- Peliculas(id, titulo, director)
- ActuaEn(idActor, idPelicula)

Símbolo	Operación	Qué hace	Ejemplo
π	Proyección	Selecciona columnas (atributos).	$\pi_{nombre}(Actores)$
σ	Selección	Filtra filas (tuplas) por condición.	$\sigma_{edad<30}(Actores)$
$\cup$	Unión	Une dos relaciones con el mismo esquema.	$\pi_{nombre}(Actores) \cup \pi_{director}(Peliculas)$
$-$	Diferencia	Devuelve tuplas en una relación que no están en la otra.	$\pi_{nombre}(Actores) - \pi_{nombre}(Actores \bowtie ActuaEn)$
$\cap$	Intersección	Devuelve tuplas que están en ambas relaciones.	$\pi_{nombre}(Actores) \cap \pi_{director}(Peliculas)$
$\times$	Producto cruz	Combina todas las tuplas de una relación con todas las de otra.	$Actores \times Peliculas$
$\bowtie$	Join (con condición)	Une dos relaciones por un atributo clave.	$Actores \bowtie_{id=idActor} ActuaEn$
$\bowtie$ (natural)	Natural Join	Une por atributos con el mismo nombre (sin condición explícita).	$Actores \bowtie ActuaEn$
ρ	Renombrar	Cambia nombre de atributos o de la relación.	$\rho((nombre \rightarrow actorNombre), Actores)$

Notas: π proyecta atributos; σ aplica predicados; $\cup, \cap, -$ requieren esquemas compatibles; $\times$ es producto cartesiano; $\bowtie$ con subíndice indica condición ($\bowtie_{\varphi}$); el natural join usa coincidencia de nombres; ρ renombra relación y/o atributos.

Pregunta 1 - Álgebra Relacional

Sean las siguientes relaciones, que contienen información sobre **Jugadores**, **Rankings**, **Torneos** e **Inscripciones** de tenis. A continuación se muestran **ejemplos** de tuplas en las 4 tablas:

id_jugador	nombre	pais	edad
1	Novak Djokovic	Serbia	37
2	Carlos Alcaraz	España	21
3	Alejandro Tabilo	Chile	27
...	...	...	...
n	Luca Fernandini	Italia	23

Cuadro 1: Jugadores

id_jugador	ranking_singles	ranking_dobles
1	5	491
2	3	519
3	32	118
...	...	...
n	1	164

Cuadro 2: Rankings

id_torneo	nombre_torneo	ciudad	superficie	puntos
1	Wimbledon	Londres	pasto	2000
2	Roland Garros	Paris	arcilla	2000
3	Indian Wells	California	cemento	1000
...	...	...	...	...
m	Chile Open	Santiago	arcilla	250

Cuadro 3: Torneos

id_inscripcion	id_jugador	id_torneo	fecha_inscripcion
1	1	1	03-06-2025
2	3	1	12-06-2025
3	3	3	09-08-2025
...	...	...	...
k	n	m	31-01-2025

Cuadro 4: Inscripciones

Escriba las siguientes operaciones en álgebra relacional, utilizando la sintaxis vista en clases (puede encontrarla en el torpedo):

Hint: renombre las relaciones a J, R, T, I, respectivamente, con el operador correspondiente.

a) (3 pts) Liste el nombre y país de los jugadores mayores de 35 años con un ranking en dobles menor a 50.

b) (4 pts) Liste el nombre y fecha de inscripción de todos los jugadores que estén inscritos en torneos realizados en la ciudad de París.

id_jugador	nombre	pais	edad
1	Novak Djokovic	Serbia	37
2	Carlos Alcaraz	España	21
3	Alejandro Tabilo	Chile	27
...	...	...	...
n	Luca Fernandini	Italia	23

Cuadro 1: Jugadores

id_torneo	nombre_torneo	ciudad	superficie	puntos
1	Wimbledon	Londres	pasto	2000
2	Roland Garros	Paris	arcilla	2000
3	Indian Wells	California	cemento	1000
...	...	...	...	...
m	Chile Open	Santiago	arcilla	250

Cuadro 3: Torneos

id_jugador	ranking_singles	ranking_dobles
1	5	491
2	3	519
3	32	118
...	...	...
n	1	164

Cuadro 2: Rankings

id_inscripcion	id_jugador	id_torneo	fecha_inscripcion
1	1	1	03-06-2025
2	3	1	12-06-2025
3	3	3	09-08-2025
...	...	...	...
k	n	m	31-01-2025

Cuadro 4: Inscripciones

Escriba las siguientes operaciones en álgebra relacional, utilizando la sintaxis vista en clases (puede encontrarla en el torpedo):

Hint: renombre las relaciones a J, R, T, I, respectivamente, con el operador correspondiente.

a) (3 pts) Liste el nombre y país de los jugadores mayores de 35 años con un ranking en dobles menor a 50.

b) (4 pts) Liste el nombre y fecha de inscripción de todos los jugadores que estén inscritos en torneos realizados en la ciudad de París.

a) (3 pts) Liste el nombre y país de los jugadores mayores de 35 años con un ranking en dobles menor a 50.

Solución:

Se renombran las relaciones por comodidad:

$$\rho(J,\ Jugadores), \quad \rho(R,\ Rankings), \quad \rho(T,\ Torneos), \quad \rho(I,\ Inscripciones)$$

$$\pi_{nombre,pais}\left(\sigma_{edad>35 \wedge ranking_dobles<50}(J \bowtie_{J.id_jugador=R.id_jugador} R)\right)$$

b) (4 pts) Liste el nombre y fecha de inscripción de todos los jugadores que estén inscritos en torneos realizados en la ciudad de París.

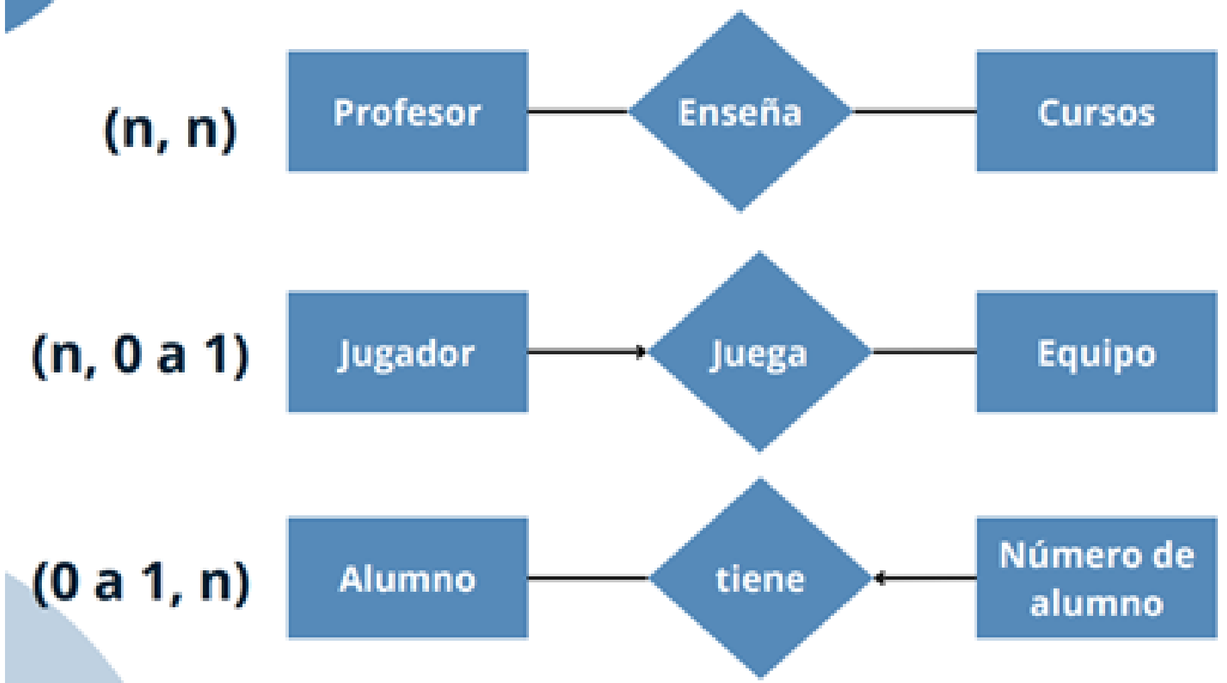
Solución:

$$\pi_{nombre,fecha_inscripcion}\left(J \bowtie_{J.id_jugador=I.id_jugador} I \bowtie_{I.id_torneo=T.id_torneo} \left(\sigma_{ciudad='Paris'} T\right)\right)$$

Modelo Entidad/Relación y Paso a Tablas

Modelo Entidad/Relación

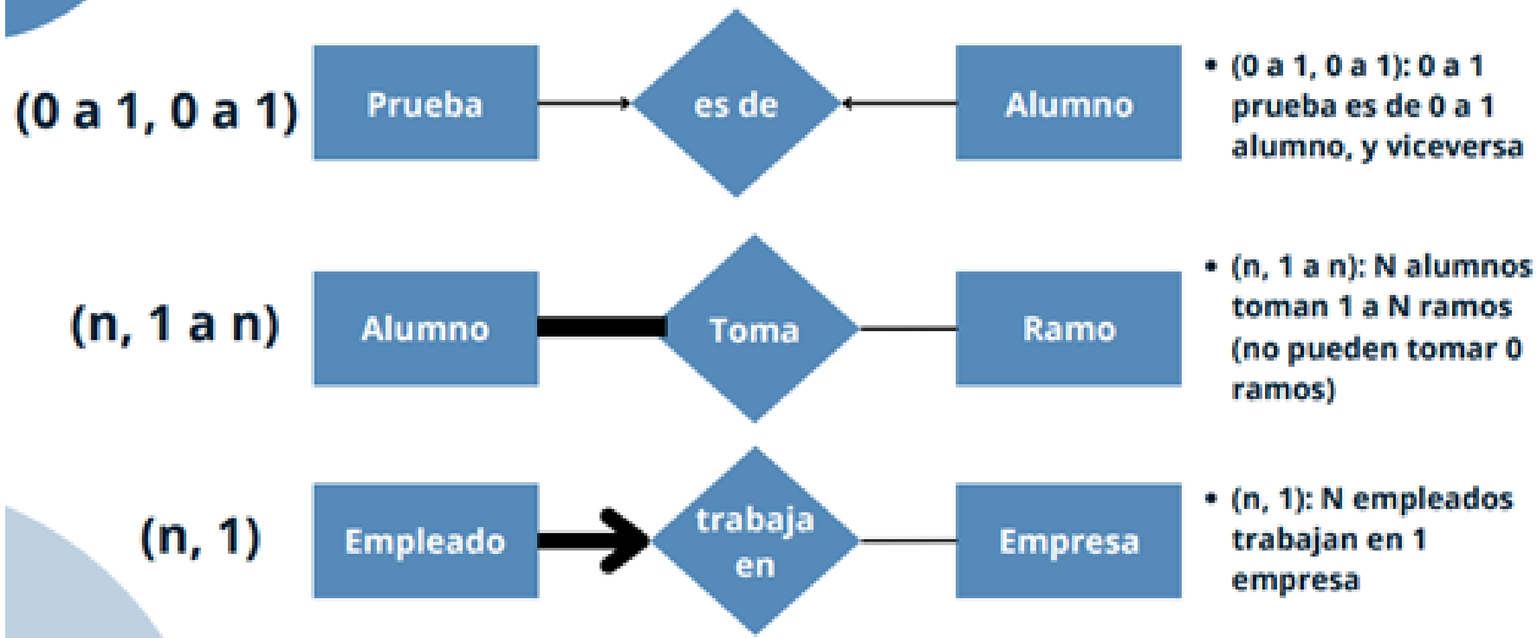
Relaciones Binarias



- (n, n): N profesor enseña en N cursos. N curso es enseñado por N profesores.
- (n, 0 a 1): N jugadores juegan por 0 o 1 equipo. En 1 equipo jugan N jugadores
- (0 a 1, n): 0 a 1 alumno tiene N num de alumnos. 1 num de alumno es de 0 a 1 alumno.

Modelo Entidad/Relación

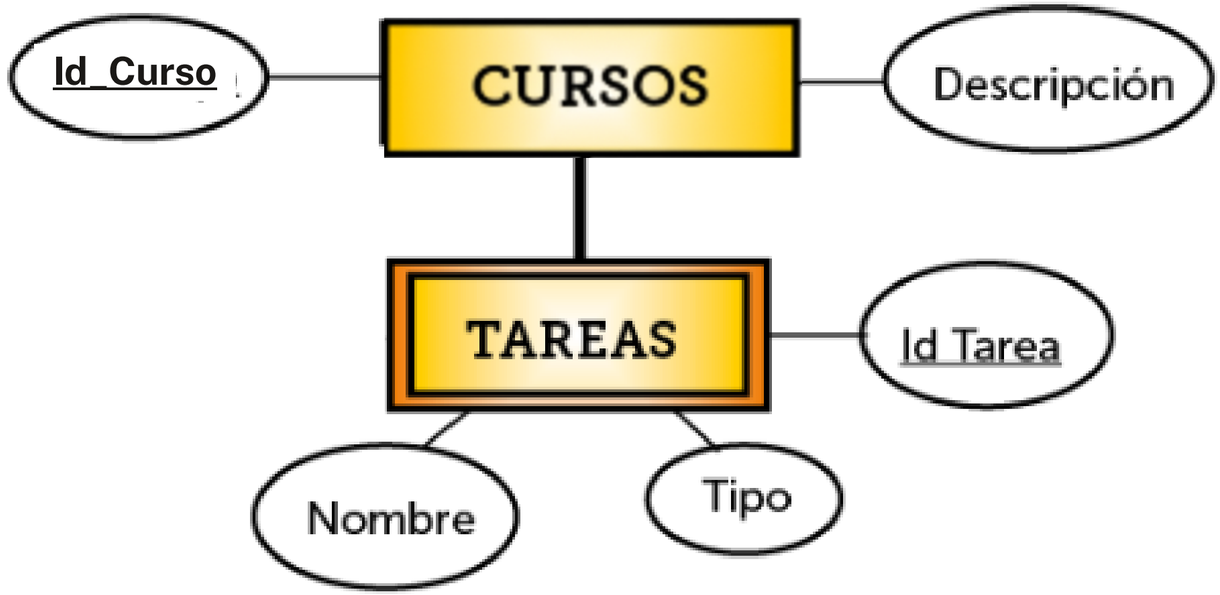
Relaciones Binarias



- (0 a 1, 0 a 1): 0 a 1 prueba es de 0 a 1 alumno, y viceversa
- (n, 1 a n): N alumnos toman 1 a N ramos (no pueden tomar 0 ramos)
- (n, 1): N empleados trabajan en 1 empresa

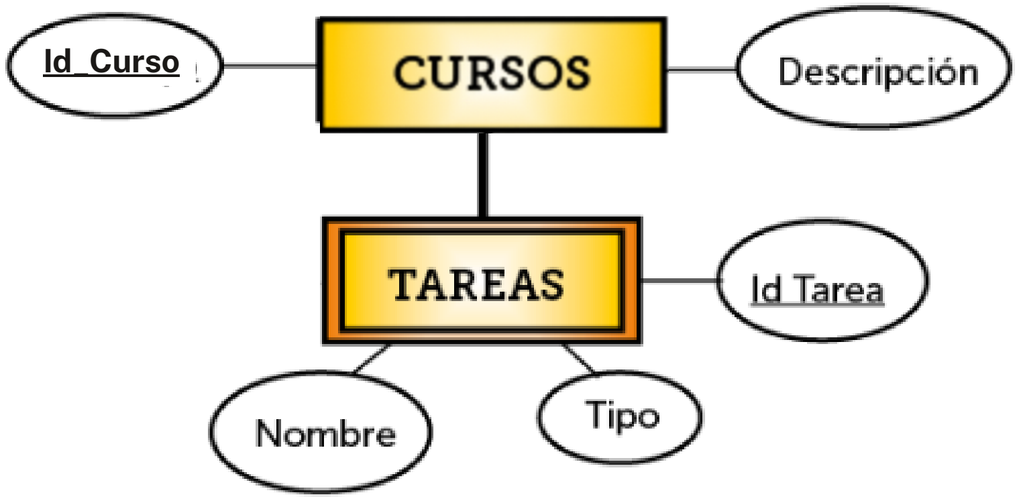
Paso a Tablas

1. Entidades fuertes → tablas
2. Entidades débiles → tablas (incluye claves foráneas de las entidades padres)



Paso a Tablas

1. Entidades fuertes → tablas
2. Entidades débiles → tablas (incluye claves foráneas de las entidades padres)

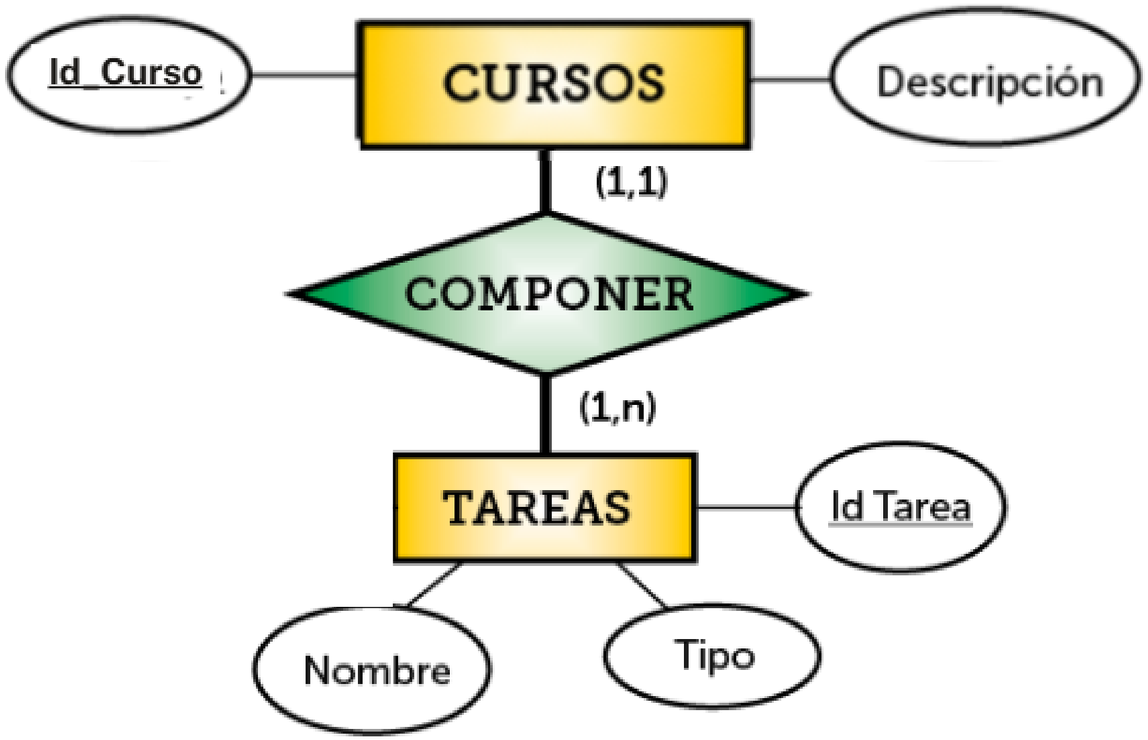


Cursos(Id_Curso, Descripción)

Tareas(Id_Tarea, Id_Curso , Nombre, Tipo)

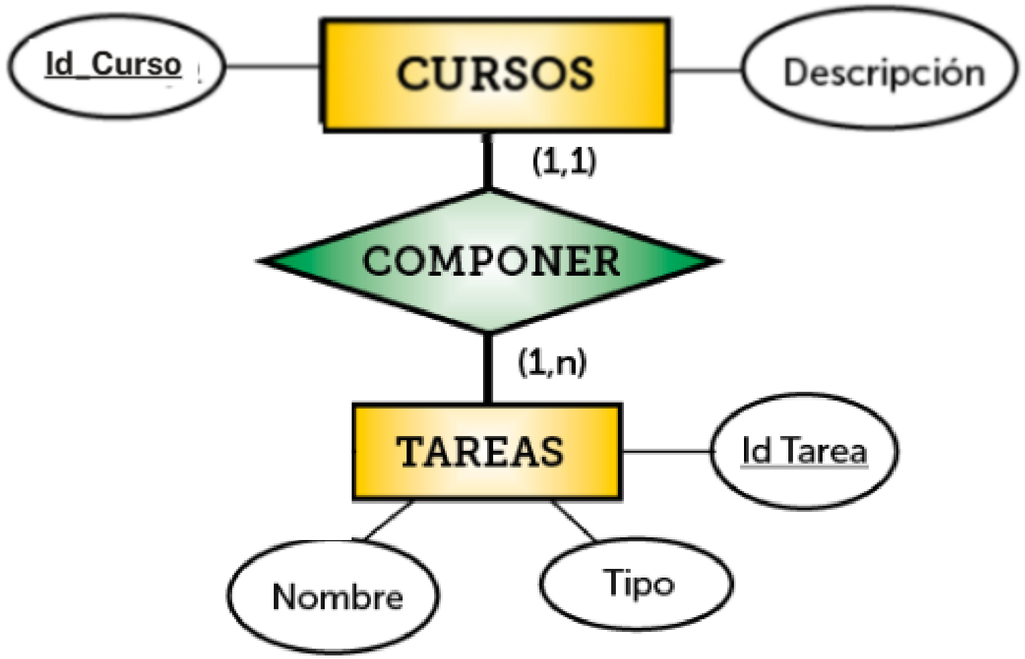
Paso a Tablas

3. Relaciones 1:N sin atributos→ clave foránea en la tabla "muchos")



Paso a Tablas

3. Relaciones 1:N sin atributos→ clave foránea en la tabla "muchos")

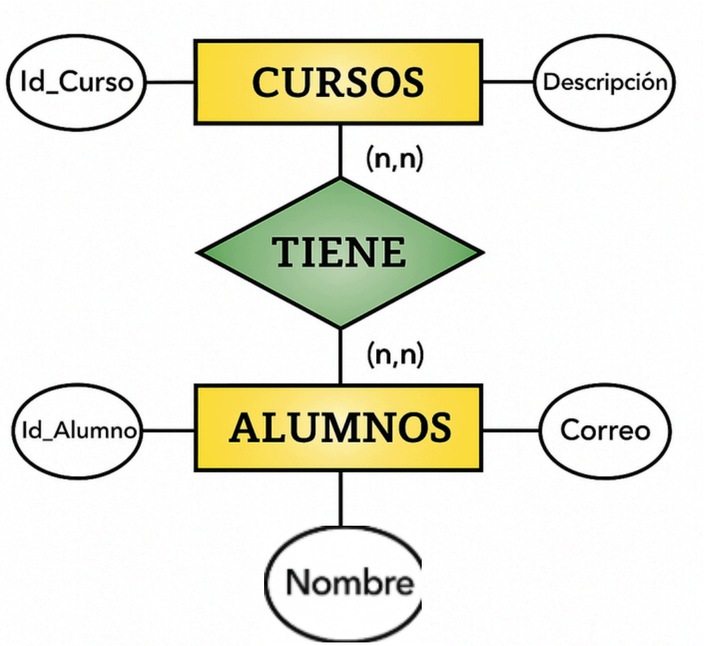


Cursos(Id_Curso, Descripción)

Tareas(Id_Tarea , Nombre, Tipo, Id_Curso FK)

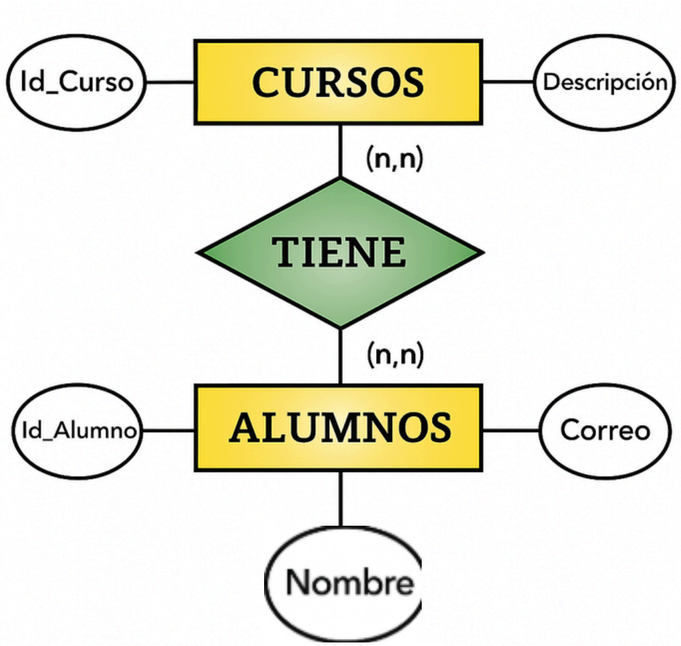
Paso a Tablas

4. Relaciones N:M con o sin atributos→ tabla intermedia que incluye llave foránea de las entidades relacionadas.



Paso a Tablas

4. Relaciones N:M con o sin atributos→ tabla intermedia que incluye llave foránea de las entidades relacionadas.



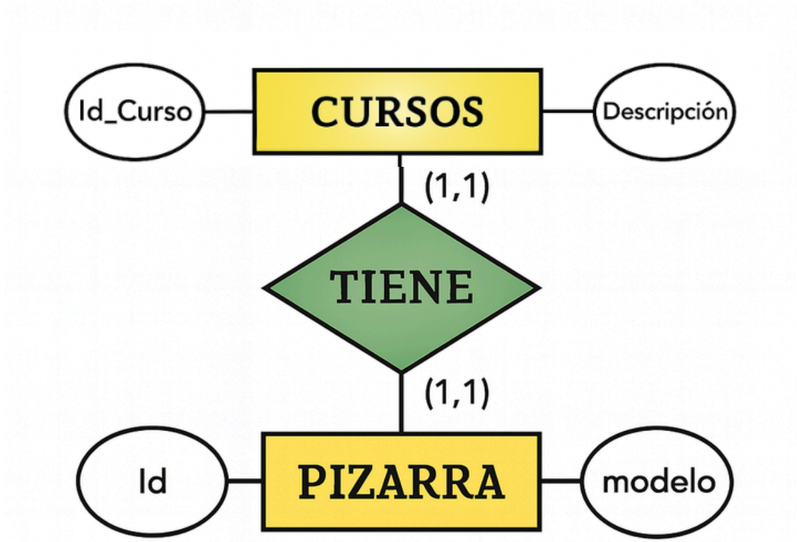
Cursos(Id_Curso, Descripción)

Alumnos(Id_Alumno, Nombre,)

Tiene (Id_Curso, Id_Alumno)

Paso a Tablas

5. Relaciones 1:1 sin atributos → cualquiera de las dos entidades absorbe llave foránea de la otra entidad

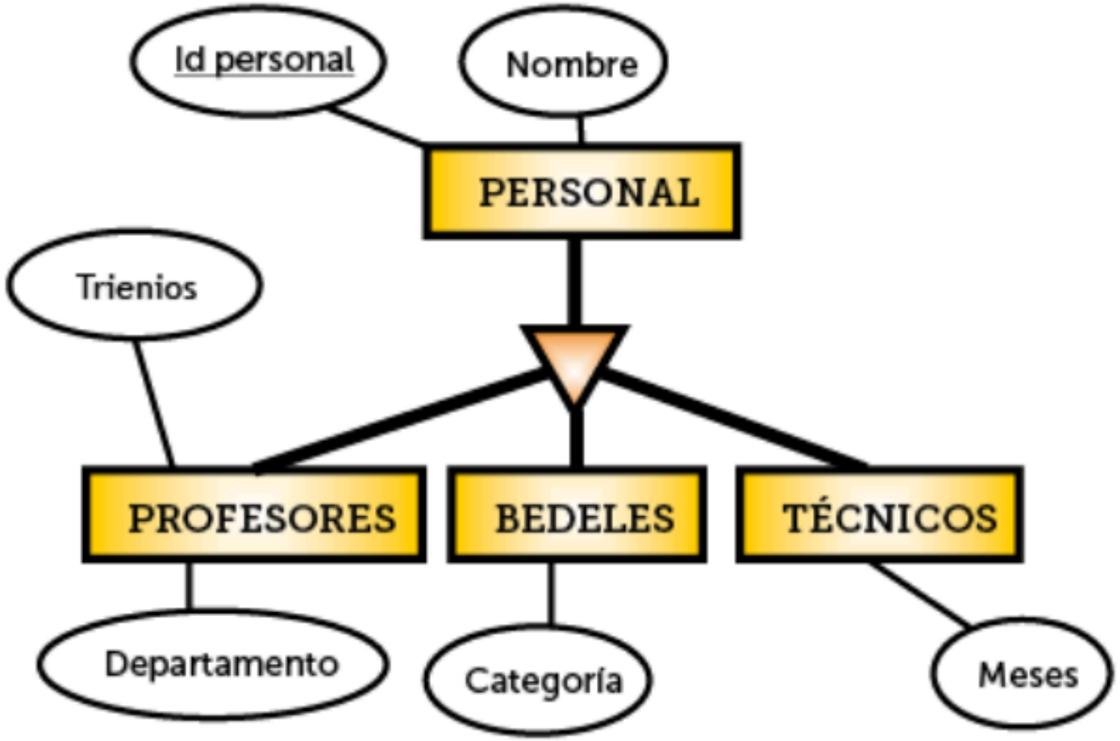


Cursos(Id_Curso, Descripción)
Pizarra(Id , modelo, Id_Curso FK)

Cursos(Id_Curso, Descripción, Id_Pizarra FK)
Pizarra(Id , modelo)

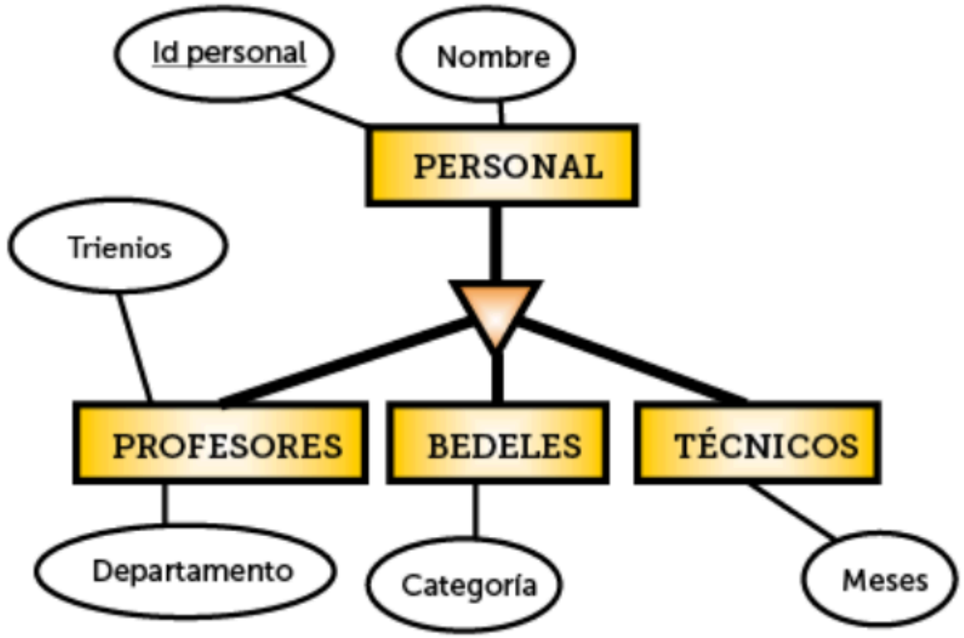
Paso a Tablas

6. Relaciones “IsA” → Heredar id del padre.



Paso a Tablas

6. Relaciones “IsA” → Heredar id del padre.



Personal (Id_personal, Nombre)

Profesores (Id_personal, Trienios, Departamento)

Bedeles (Id_personal, Categoría)

Técnicos (Id_personal, meses)

Resumen Paso a Tablas

1. Entidades fuertes → tablas

2. Entidades débiles → tablas (incluyendo claves foráneas de las entidades fuertes)

3. Relaciones 1:N sin atributos→ clave foránea en la tabla "muchos"

4. Relaciones N:M con o sin atributos→ tabla intermedia que incluye llave foránea de las entidades relacionadas.

5. Relaciones 1:1 sin atributos → cualquiera de las dos entidades absorbe llave foránea de la otra entidad

6. Relaciones “IsA” → Heredar id del padre.

Pregunta 2 - Diagrama E/R y Esquema Relacional

Considere la red social de música DCCanciones (inspirada en Spotify), en la que los usuarios pueden seguir a otros usuarios y también interactuar con contenido musical.

Cada usuario se identifica por un `id_usuario` y tiene atributos nombre, usuario, correo electrónico, canción favorita, y tipo. Los tipos de usuario pueden ser básico o premium. Los usuarios pueden seguir a otros usuarios dentro de la plataforma. Para esto, deben enviar una solicitud de seguimiento a otro usuario, la cual puede ser aceptada o rechazada posteriormente. Esta se registra con una fecha en la que se realizó la solicitud, y su estado actual.

Los usuarios pueden crear listas de reproducción (playlists), las que contienen canciones. Cada lista de reproducción tiene un nombre, una serie de canciones, una fecha de creación y está asociada a un único creador (usuario).

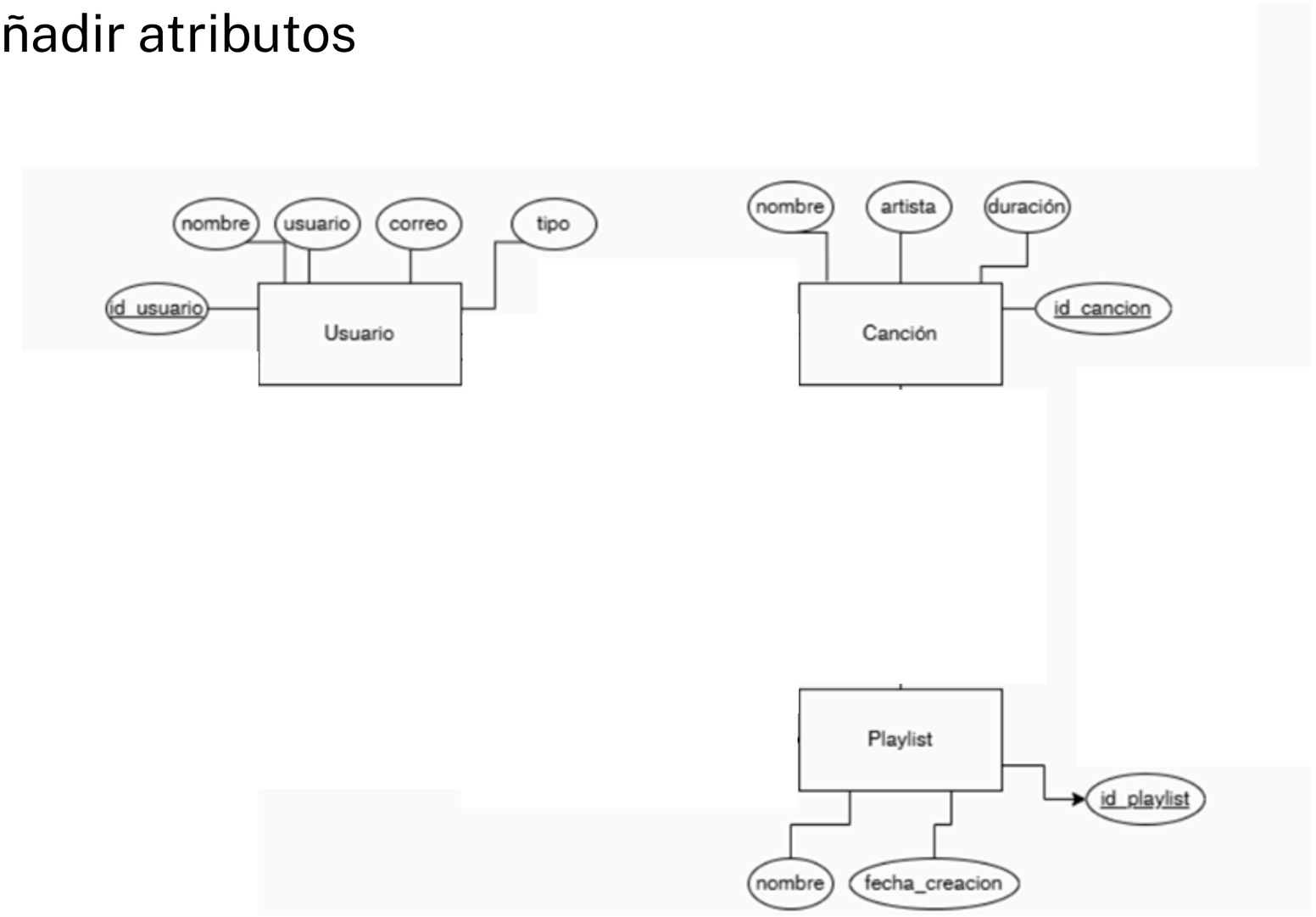
Las canciones están identificadas por un `id_cancion` y tienen atributos nombre, artista y duración. Una misma canción puede estar incluida en múltiples playlists.

a) (10 pts) Diseñe el Diagrama Entidad/Relación de DCCanciones. Identifique claramente las entidades, entidades débiles, relaciones, cardinalidades, atributos, y llaves primarias y parciales.

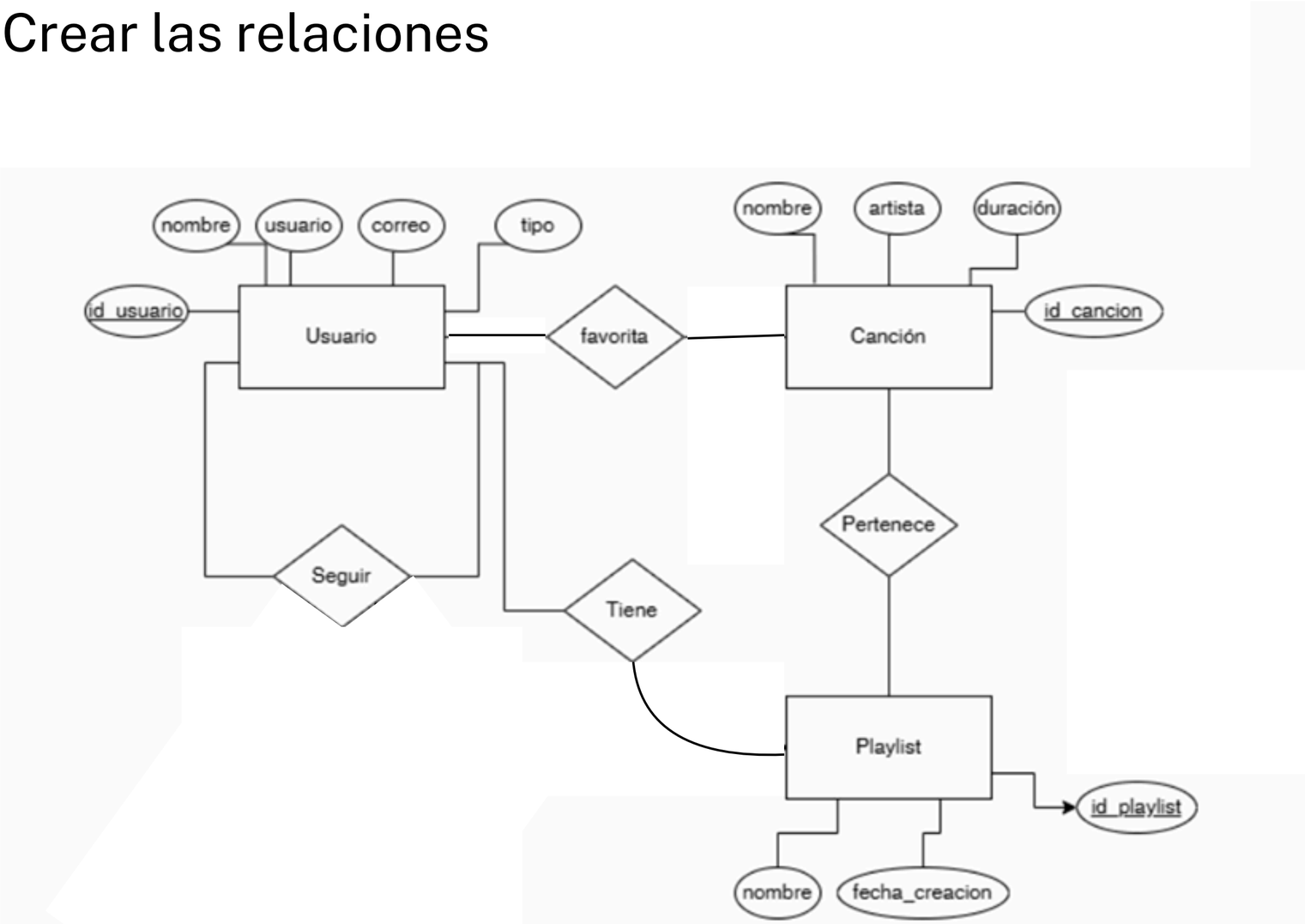
1. Identificar las Entidades



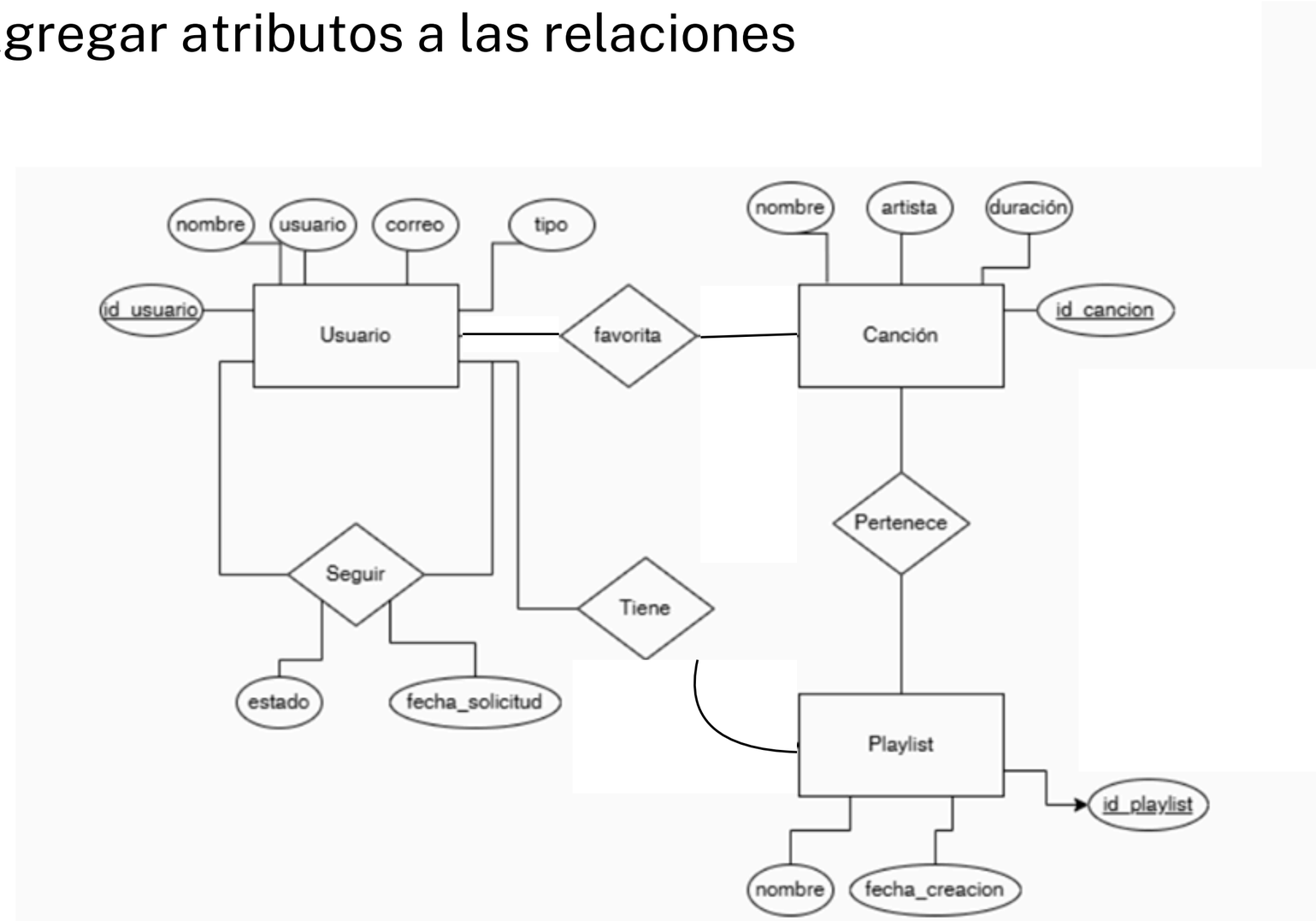
2. Añadir atributos



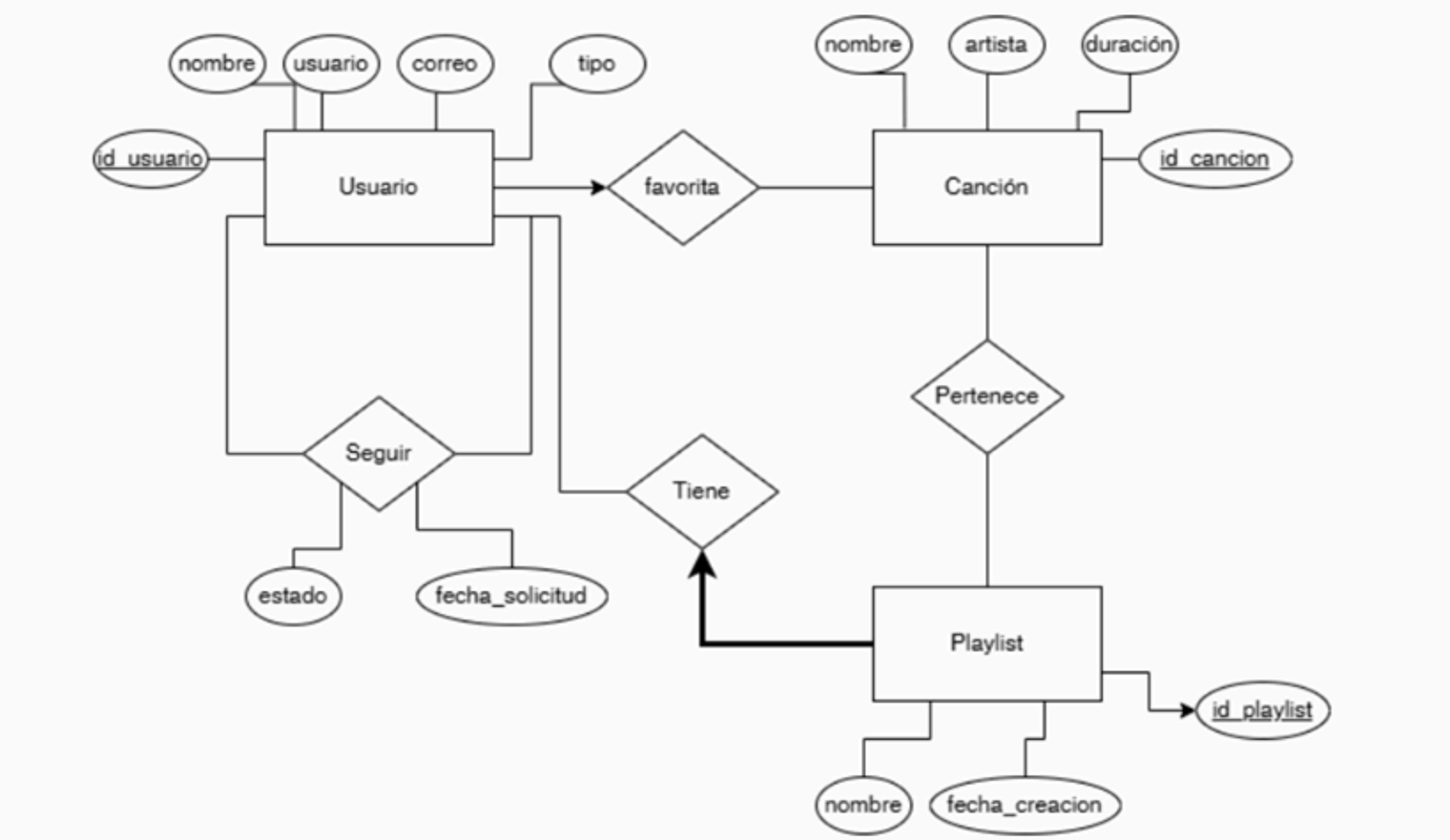
3. Crear las relaciones



4. Agregar atributos a las relaciones

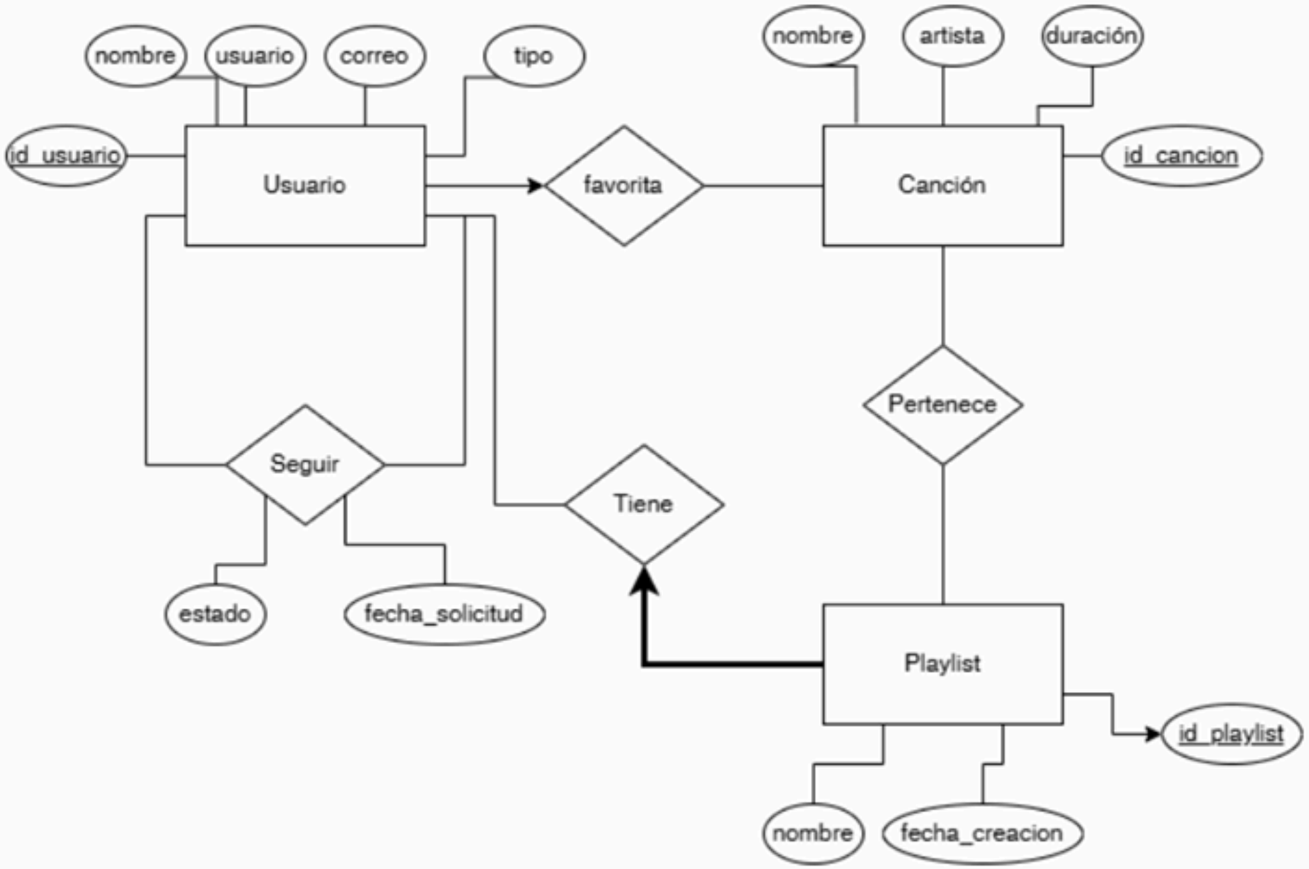


5. Definir cardinalidad



a) (10 pts) Diseñe el Diagrama Entidad/Relación de DCCanciones. Identifique claramente las entidades, entidades débiles, relaciones, cardinalidades, atributos, y llaves primarias y parciales.

Solución:



b) (5 pts) A partir del Diagrama E/R propuesto, genere el Esquema Relacional asociado.

Para cada tabla, indique los atributos, sus dominios, las llaves primarias, foráneas, y cualquier otra aclaración respecto a la representación de sus relaciones.

b) (5 pts) A partir del Diagrama E/R propuesto, genere el Esquema Relacional asociado.

Para cada tabla, indique los atributos, sus dominios, las llaves primarias, foráneas, y cualquier otra aclaración respecto a la representación de sus relaciones.

```
CREATE TABLE Usuario (
    id_usuario INT PRIMARY KEY,
    nombre VARCHAR(50),
    usuario VARCHAR(30),
    correo VARCHAR(100),
    cancion_favorita INT,
    tipo VARCHAR(10),
    FOREIGN KEY (cancion_favorita) REFERENCES Cancion(id_cancion)
);

CREATE TABLE Cancion (
    id_cancion INT PRIMARY KEY,
    nombre VARCHAR(100),
    artista VARCHAR(100),
    duracion INT -- duración en segundos
);

CREATE TABLE Playlist (
    id_playlist INT PRIMARY KEY,
    nombre VARCHAR(100),
    fecha_creacion DATE,
    creador INT,
    FOREIGN KEY (creador) REFERENCES Usuario(id_usuario)
);
```

```
CREATE TABLE Seguir (
    id_origen INT,
    id_destino INT,
    estado VARCHAR(10), -- pendiente, aceptada, rechazada
    fecha_solicitud DATE,
    PRIMARY KEY (id_origen, id_destino),
    FOREIGN KEY (id_origen) REFERENCES Usuario(id_usuario),
    FOREIGN KEY (id_destino) REFERENCES Usuario(id_usuario)
);

CREATE TABLE Pertenece (
    id_playlist INT,
    id_cancion INT,
    PRIMARY KEY (id_playlist, id_cancion),
    FOREIGN KEY (id_playlist) REFERENCES Playlist(id_playlist),
    FOREIGN KEY (id_cancion) REFERENCES Cancion(id_cancion)
);
```

SQL

- **SELECT ... FROM**
  Extrae columnas específicas (SELECT) de una tabla (FROM)

- **WHERE**
  Filtra filas que cumplen una condición

- **DISTINCT**
  Elimina valores duplicados de los resultados

- **LIKE**
  Busca un patrón específico dentro de un texto (ej: LIKE ‘text’)

- **ORDER BY**
  Ordena los resultados de forma ascendente (ASC) o descendente (DESC)

- **INSERT INTO**
  Agrega una nueva fila de datos a la tabla

- **UPDATE ... SET**
  Modifica datos de filas existentes

- **DELETE FROM**
  Elimina filas de una tabla

```
SELECT DISTINCT nombre, categoria
FROM Peliculas
WHERE (categoria = 'SciFi' OR categoria =
'Adventure') AND nombre LIKE '%o%'ORDER BY
nombre ASC;
```

```
INSERT INTO Peliculas(id, nombre, categoria)
VALUES (6, 'Dune', 'SciFi');
```

```
UPDATE Peliculas SET calificacion = 8.0
WHERE id = 6;
```

```
DELETE FROM Peliculas WHERE id = 6;
```

SQL

- **GROUP BY**

Agrupa filas con el mismo valor para aplicarles una función de agregación.

Funciones de agregación

- **COUNT ()**

Cuenta el número de filas.

- **SUM ()**

Suma los valores de una columna numérica.

- **AVG ()**

Calcula el promedio de una columna numérica.

- **MAX ()/MIN ()**

Obtiene el valor máximo o mínimo de una columna.

- **HAVING**

Filtra los grupos basados en una condición de agregación (se usa después de GROUP BY).

```sql
SELECT
    director,
    COUNT(id) AS total_peliculas,
    AVG(calificacion) AS promedio_cal
FROM Peliculas
GROUP BY director
HAVING COUNT(id) > 1;
```

SQL

- **JOIN**
  Combina filas de dos tablas cuando los valores de las columnas relacionadas coinciden. Solo muestra resultados que existen en ambas tablas.
- **OUTER JOIN**
  Combina filas e incluye las que no tienen coincidencia en una de las tablas, rellenando los datos faltantes con NULL.
- **ANIDACIÓN**
  Es una consulta SELECT que se ejecuta dentro de otra consulta principal. Se usa como un filtro o como una tabla temporal.

```
SELECT
    a.nombre,
    COALESCE(agg.total_peliculas, 0) AS
peliculas_actuadas
FROM Actores AS a
LEFT JOIN (
    SELECT id_actor, COUNT(id_pelicula)
AS total_peliculas
    FROM Actuo_en
    GROUP BY id_actor
) AS agg ON a.id = agg.id_actor;
```

Formas Normales

1NF

Cada columna debe tener solo un valor, no una lista

Ejemplo tabla no normalizada:

id_alumno	nombre	curso
1	Juan	Matemáticas, Física
2	María	Química

Ejemplo tabla normalizada (1NF):

id_alumno	nombre	curso
1	Juan	Matemáticas
1	Juan	Física
2	Maria	Química

2NF

- **Debe estar en 1NF.**
- **Cada columna debe depender de la clave primaria completa, no solo de una parte**

Ejemplo tabla no normalizada (2NF):

ID_Alumno	ID_Curso	Nombre_Alumno	Nombre_Curso	Profesor
1	101	Juan	Matemáticas	Prof. Pérez
1	102	Juan	Física	Prof. López
2	103	María	Química	Prof. Díaz

"Nombre_Alumno" solo depende de "ID_Alumno", no de toda la clave primaria (ID_Alumno, ID_Curso).

ID_Alumno → Nombre_Alumno

La llave completa es ID_Alumno, ID_Curso

3NF

Una relación **R** está en **3NF** si para toda dependencia funcional no trivial X → Y, **X** es una superllave o **Y** es parte de una llave minimal.

Ejemplo tabla no normalizada (3NF):

ID_Alumno	Nombre	ID_Ciudad	Ciudad	País
1	Juan	10	Santiago	Chile
2	María	11	Lima	Perú

"Ciudad" y "País" dependen de "ID_Ciudad", la cual depende de "ID_Alumno", generando dependencia transitiva

ID_Alumno → Nombre, ID_Ciudad

ID_Ciudad → Ciudad, País

BCNF

- Debe estar en 3NF.
- Cada dependencia funcional A -> B , A debe ser llave candidata siempre

Pregunta 4 - Formas Normales

Usted ha sido contratado para analizar y rediseñar una base de datos utilizada por una empresa de transporte. Actualmente, la información sobre los envíos realizados por clientes se encuentra almacenada en una única tabla, como se muestra a continuación:

id_envío	cliente	rut_cliente	peso_kg	fecha	ciudad_entrega
1001	Sofía	15.815.461-7	10	2025-10-25	Santiago, Rancagua
1002	Paula	17.531.050-9	25	2025-11-06	Santiago, Talca, Curicó
1003	Ignacio	12.899.241-3	7	2025-03-17	Antofagasta, Calama
1004	Olivia	14.923.887-5	3	2025-01-21	Santiago, Temuco, Valdivia
1005	Consuelo	20.362.056-K	18	2025-09-08	Iquique, Arica
1006	Agustín	4.816.960-0	2	2025-08-30	Puerto Montt

Cuadro 5: Tabla `Envíos`

La columna `ciudad_entrega` contiene una lista con los nombres de las ciudades que forman parte del trayecto de entrega de cada envío. Un mismo envío puede pasar por una o más ciudades dependiendo de su destino, y el orden no importa.

A partir de la tabla anterior, responda lo siguiente:

a) (5 pts) Identifique todas las dependencias funcionales presentes en la tabla.

b) (5 pts) Indique si la tabla se encuentra en 1NF, 2NF o 3NF. Justifique su respuesta.

a) (5 pts) Identifique todas las dependencias funcionales presentes en la tabla.

Solución:

- `id_envío` → `cliente, rut_cliente, peso_kg, fecha, ciudad_entrega`
- `rut_cliente` → `cliente`

b) (5 pts) Indique si la tabla se encuentra en 1NF, 2NF o 3NF. Justifique su respuesta.

Solución:

La tabla no se encuentra en 1NF, ya que el atributo `ciudad_entrega` contiene listas de ciudades, lo que viola el principio de atomicidad de la 1FN. Por lo tanto, tampoco cumple con 2NF ni 3NF.

Creado por:

- **Matías Retamales**
- **Sofia Retamales**
- **Sebastián Salgado**