1. Primera Forma Normal (1FN):

- Todas las tablas cumplen con 1FN ya que:

- Tienen una clave primaria única identificable (usando SERIAL PRIMARY KEY o claves compuestas)

- No hay grupos repetitivos ni arrays

- Todos los atributos son atómicos (no hay valores múltiples en una misma columna)

2. Segunda Forma Normal (2FN):

- Las tablas cumplen con 2FN ya que:

- Ya están en 1FN

- No hay dependencias parciales - todos los atributos no clave dependen completamente de la clave primaria

- La tabla Performances, que tiene una clave compuesta (recording\_id, performer\_id), maneja correctamente la relación muchos-a-muchos entre Recordings y Performers

3. Tercera Forma Normal (3FN):

- Las tablas cumplen con 3FN porque:

- Ya están en 2FN

- No hay dependencias transitivas - los atributos no clave no dependen de otros atributos no clave

Análisis por tabla:

1. Companies:

CREATE TABLE Companies (

company\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

address TEXT NOT NULL,

UNIQUE(name)

);

- Normalizada correctamente:

- Clave primaria única (company\_id)

- Atributos independientes entre sí

- No hay dependencias transitivas

2. Recordings:

CREATE TABLE Recordings (

recording\_id SERIAL PRIMARY KEY,

title VARCHAR(100) NOT NULL,

musical\_category VARCHAR(50) NOT NULL,

track\_count INTEGER NOT NULL CHECK (track\_count > 0),

description TEXT,

company\_id INTEGER REFERENCES Companies(company\_id),

UNIQUE(title)

);

- Normalizada correctamente:

- Clave primaria única (recording\_id)

- Referencia a Companies mediante clave foránea

- Los atributos dependen únicamente de la clave primaria

3. Performers:

CREATE TABLE Performers (

performer\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

biography TEXT,

UNIQUE(name)

);

- Normalizada correctamente:

- Clave primaria única (performer\_id)

- Atributos dependen directamente de la clave primaria

4. Performances (tabla de relación):

CREATE TABLE Performances (

recording\_id INTEGER REFERENCES Recordings(recording\_id),

performer\_id INTEGER REFERENCES Performers(performer\_id),

participation\_date DATE NOT NULL,

PRIMARY KEY (recording\_id, performer\_id)

);

- Normalizada correctamente:

- Maneja la relación muchos-a-muchos entre Recordings y Performers

- Incluye la fecha de participación como atributo adicional

- Clave primaria compuesta adecuada

5. Formats:

CREATE TABLE Formats (

format\_id SERIAL PRIMARY KEY,

name VARCHAR(50) NOT NULL,

UNIQUE(name)

);

- Normalizada correctamente:

- Clave primaria única (format\_id)

- Atributo name depende únicamente de la clave primaria

6. PhysicalCopies:

CREATE TABLE PhysicalCopies (

copy\_id SERIAL PRIMARY KEY,

recording\_id INTEGER REFERENCES Recordings(recording\_id),

format\_id INTEGER REFERENCES Formats(format\_id),

conservation\_state VARCHAR(20) CHECK (conservation\_state IN ('bueno', 'malo', 'regular'))

);

- Normalizada correctamente:

- Maneja la relación entre grabaciones y formatos

- Incluye el estado de conservación como atributo

- Referencias correctas mediante claves foráneas

El diseño evita:

- Redundancia de datos

- Anomalías de inserción, actualización y eliminación

- Dependencias parciales y transitivas

La estructura permite:

- Mantener la integridad referencial

- Realizar consultas eficientes

- Escalar la base de datos según sea necesario