CLASE 3

Bases de Datos Relacionales

Proyecto Netflix & Pokémon

SQL 1025

Práctica con Relaciones Complejas

En esta clase trabajarás con:

- Base de datos completa de Netflix con múltiples tablas
- Relaciones uno a muchos y muchos a muchos
- Inserción masiva de datos realistas
- Consultas complejas con JOINs
- Proyecto final: Base de datos Pokémon

Índice

1 Introducción

Objetivos de la Clase

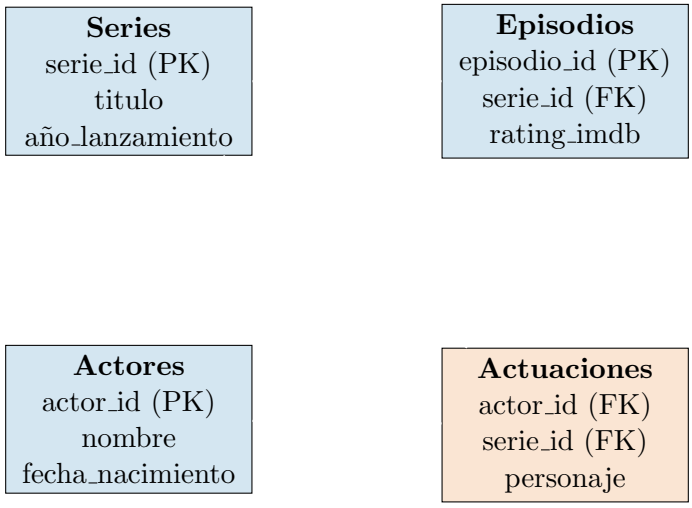
Al finalizar esta clase, serás capaz de:

- ✔ Crear bases de datos con múltiples tablas interrelacionadas
- ✔ Implementar relaciones uno a muchos y muchos a muchos
- ✔ Insertar grandes volúmenes de datos de manera eficiente
- ✔ Utilizar tipos de datos especializados (DECIMAL, TEXT, DATE)
- ✔ Diseñar bases de datos basadas en requisitos del mundo real

2 Proyecto Netflix: Base de Datos Completa

En esta sección crearemos una base de datos completa que simula el sistema de información de Netflix, incluyendo series, episodios, actores y sus actuaciones.

2.1 Diagrama de la Base de Datos



Concepto Clave

Tipos de Relaciones:

- **Uno a Muchos (1:N):** Una serie puede tener muchos episodios
- **Muchos a Muchos (N:N):** Un actor puede actuar en muchas series, y una serie puede tener muchos actores (implementado a través de la tabla Actuaciones)

2.2 Crear la Base de Datos

```
1 CREATE DATABASE IF NOT EXISTS NetflixDB;
2 USE NetflixDB;
```

Nota Importante

La cláusula `IF NOT EXISTS` evita errores si la base de datos ya existe. Es una buena práctica para scripts que se ejecutan múltiples veces.

2.3 Creación de Tablas

2.3.1 Tabla Series

Esta tabla almacena información básica sobre cada serie:

```
CREATE TABLE IF NOT EXISTS Series (
    serie_id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    descripcion TEXT,
    a o_lanzamiento INT,
    genero VARCHAR(255)
);
```

Análisis de la estructura:

- `serie_id`: Clave primaria con auto-incremento
- `titulo`: Campo obligatorio (`NOT NULL`)
- `descripcion`: Tipo `TEXT` para contenido largo
- `año_lanzamiento`: Tipo `INT` para el año
- `genero`: Categoría de la serie

2.3.2 Tabla Episodios

Almacena los episodios de cada serie con información detallada:

```
CREATE TABLE IF NOT EXISTS Episodios (
    episodio_id INT AUTO_INCREMENT PRIMARY KEY,
    serie_id INT,
    titulo VARCHAR(255) NOT NULL,
    duracion INT,
    rating_imdb DECIMAL(3,1),
    temporada INT,
    descripcion TEXT,
    fecha_estreno DATE,
    FOREIGN KEY (serie_id) REFERENCES Series(serie_id)
);
```

Concepto Clave

Tipo de dato DECIMAL(3,1):
Este tipo de dato permite almacenar números decimales con precisión:

- **3**: Número total de dígitos
- **1**: Número de decimales
- Ejemplos válidos: 9.5, 10.0, 8.7

Perfecto para ratings de IMDb que van de 0.0 a 10.0.

2.3.3 Tabla Actores

Información sobre los actores:

```
CREATE TABLE IF NOT EXISTS Actores (
    actor_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    fecha_nacimiento DATE
);
```

2.3.4 Tabla Actuaciones (Relación Muchos a Muchos)

Esta es una tabla de unión que relaciona actores con series:

```
CREATE TABLE IF NOT EXISTS Actuaciones (
    actor_id INT,
    serie_id INT,
    personaje VARCHAR(255),
    FOREIGN KEY (actor_id) REFERENCES Actores(actor_id),
    FOREIGN KEY (serie_id) REFERENCES Series(serie_id),
    PRIMARY KEY (actor_id, serie_id)
);
```

Concepto Clave

Clave Primaria Compuesta:
`PRIMARY KEY (actor_id, serie_id)` significa que la combinación de estos dos campos debe ser única. Un actor puede aparecer en una serie solo una vez, pero puede actuar en múltiples series diferentes.

3 Inserción de Datos en NetflixDB

3.1 Insertar Series

Vamos a insertar 13 series populares:

```
USE NetflixDB;

INSERT INTO Series (titulo, descripcion, a o_lanzamiento, genero)
VALUES
    ('Breaking Bad',
     'Un profesor de quimica se convierte en un rey del narcotrafico.
        ',
     2008, 'Drama'),
    ('Stranger Things',
     'Ni os en los 80s enfrentan fuerzas sobrenaturales y secretos
        gubernamentales.',
     2016, 'Ciencia ficcion'),
    ('The Crown',
     'Drama historico sobre el reinado de la Reina Isabel II del
        Reino Unido.',
     2016, 'Drama historico'),
    ('Black Mirror',
     'Serie de antologia que explora un futuro inquietante y
        distopico.',
     2011, 'Ciencia ficcion'),
    ('The Witcher',
     'Un cazador de monstruos lucha por encontrar su lugar en un
        mundo donde las personas a menudo son mas perversas.',
     2019, 'Fantasia'),
    ('The Mandalorian',
     'Un pistolero solitario explora los confines de la galaxia,
        lejos de la autoridad de la Nueva Republica.',
     2019, 'Ciencia ficcion'),
    ('BoJack Horseman',
     'Un caballo antropomorfico lucha con la depresion y la adiccion
        en esta comedia de animacion para adultos.',
     2014, 'Comedia'),
    ('Arcane',
     'Basada en el universo de League of Legends, esta serie explora
        los origenes de algunos campeones iconicos.',
     2021, 'Animacion'),
    ('Peaky Blinders',
     'Una banda de gansteres de Birmingham de la decada de 1920 se
        eleva a la prominencia.',
     2013, 'Drama historico'),
    ('Sherlock',
     'El detective mas famoso del mundo resuelve misterios en el
        Londres del siglo XXI.',
     2010, 'Drama'),
    ('Narcos',
     'La historia del narcotrafico en Colombia',
     2015, 'Biografia'),
    ('Game of Thrones',
     'Nobles familias luchan por el control del Trono de Hierro',
     2011, 'Fantasia'),
    ('The Office',
     'La vida cotidiana de los empleados de Dunder Mifflin',
     2005, 'Comedia');
```

Nota Importante

Al insertar múltiples filas, es más eficiente usar una sola sentencia `INSERT` con múltiples `VALUES` en lugar de múltiples sentencias `INSERT` separadas.

3.2 Insertar Actores

Insertamos 35 actores famosos:

```sql
INSERT INTO Actores (nombre, fecha_nacimiento) VALUES
    ('Bryan Cranston', '1956-03-07'),
    ('Millie Bobby Brown', '2004-02-19'),
    ('Claire Foy', '1984-04-16'),
    ('Mads Mikkelsen', '1965-11-22'),
    ('Henry Cavill', '1983-05-05'),
    ('Pedro Pascal', '1975-04-02'),
    ('Will Arnett', '1970-05-04'),
    ('Hailee Steinfeld', '1996-12-11'),
    ('Cillian Murphy', '1976-05-25'),
    ('Benedict Cumberbatch', '1976-07-19'),
    ('Wagner Moura', '1976-06-27'),
    ('Emilia Clarke', '1986-10-23'),
    ('Steve Carell', '1962-08-16'),
    ('Aaron Paul', '1979-08-27'),
    ('Winona Ryder', '1971-10-29'),
    ('Olivia Colman', '1974-01-30'),
    ('Anya Chalotra', '1996-07-21'),
    ('Gina Carano', '1982-04-16'),
    ('Amy Sedaris', '1961-03-29'),
    ('Sophie Turner', '1996-02-21');
-- Y mas actores...
```

Ejemplo Práctico

Formato de fechas en MySQL:
Las fechas se escriben en formato `'YYYY-MM-DD'`:

- `'1956-03-07'` = 7 de marzo de 1956
- `'2004-02-19'` = 19 de febrero de 2004

3.3 Insertar Episodios (Ejemplo con Breaking Bad)

Debido a la extensión, mostramos un ejemplo con algunos episodios de Breaking Bad:

```sql
INSERT INTO Episodios
    (serie_id, titulo, duracion, rating_imdb, temporada,
     descripcion, fecha_estreno)
VALUES
    (1, 'Piloto', 58, 9, 1,
     'Un profesor de quimica recibe un diagnostico terminal y decide
        convertirse en fabricante de metanfetaminas.',
     '2008-01-20'),
    (1, 'Gato en un saco', 47, 8.7, 2,
```

```
9       'Walt y Jesse tratan de deshacerse de dos cuerpos mientras que
            al mismo tiempo tienen que lidiar con un testigo problematico
            .',
10      '2009-05-10'),
11     (1, 'Mandala', 48, 8.6, 2,
12      'Walt y Jesse deciden buscar un nuevo socio para poder expandir
            su negocio.',
13      '2009-05-17'),
14     (1, 'Phoenix', 47, 9.1, 2,
15      'Walt experimenta uno de sus mayores conflictos eticos cuando
            Jesse hace una nueva amiga.',
16      '2009-05-24'),
17     (1, 'ABQ', 47, 9.2, 2,
18      'Desastres colaterales llevan a Walt a manipular eventos que
            afectan a los que lo rodean.',
19      '2009-05-31');
20 -- Continuar con mas episodios...
```

Nota Importante

En el material completo se incluyen más de 150 episodios de las diferentes series. Para esta demostración, mostramos la estructura básica.

3.4 Insertar Actuaciones

Relacionamos actores con sus series y personajes:

```
1  INSERT INTO Actuaciones (actor_id, serie_id, personaje) VALUES
2      (1, 1, 'Walter White'),          -- Bryan Cranston en Breaking Bad
3      (14, 1, 'Jesse Pinkman'),        -- Aaron Paul en Breaking Bad
4      (2, 2, 'Eleven'),                -- Millie Bobby Brown en Stranger
           Things
5      (15, 2, 'Joyce Byers'),          -- Winona Ryder en Stranger Things
6      (3, 3, 'Reina Isabel II'),       -- Claire Foy en The Crown
7      (5, 5, 'Geralt de Rivia'),       -- Henry Cavill en The Witcher
8      (17, 5, 'Yennefer de Vengerberg'), -- Anya Chalotra en The
           Witcher
9      (6, 6, 'Din Djarin'),            -- Pedro Pascal en The Mandalorian
10     (9, 9, 'Tommy Shelby'),          -- Cillian Murphy en Peaky
           Blinders
11     (10, 10, 'Sherlock Holmes'),     -- Benedict Cumberbatch en
           Sherlock
12     (12, 12, 'Daenerys Targaryen'), -- Emilia Clarke en Game of
           Thrones
13     (13, 13, 'Michael Scott');       -- Steve Carell en The Office
```

4 Consultas Útiles en NetflixDB

4.1 Consultas Básicas

4.1.1 Ver todas las series

```sql
SELECT * FROM Series;
```

4.1.2 Ver series de un género específico

```sql
SELECT titulo, a o_lanzamiento, genero
FROM Series
WHERE genero = 'Ciencia ficcion';
```

4.1.3 Ver episodios con rating superior a 9.0

```sql
SELECT titulo, rating_imdb, temporada
FROM Episodios
WHERE rating_imdb > 9.0
ORDER BY rating_imdb DESC;
```

4.2 Consultas con JOIN

4.2.1 Ver episodios con el nombre de su serie

```sql
SELECT
    s.titulo AS Serie,
    e.titulo AS Episodio,
    e.temporada,
    e.rating_imdb
FROM Episodios e
INNER JOIN Series s ON e.serie_id = s.serie_id
WHERE e.rating_imdb >= 9.0
ORDER BY e.rating_imdb DESC;
```

4.2.2 Ver actores y sus personajes

```sql
SELECT
    a.nombre AS Actor,
    s.titulo AS Serie,
    act.personaje AS Personaje
FROM Actuaciones act
INNER JOIN Actores a ON act.actor_id = a.actor_id
INNER JOIN Series s ON act.serie_id = s.serie_id
ORDER BY s.titulo, a.nombre;
```

4.2.3 Contar episodios por serie

```sql
SELECT
    s.titulo AS Serie,
    COUNT(e.episodio_id) AS Total_Episodios,
    AVG(e.rating_imdb) AS Rating_Promedio
FROM Series s
LEFT JOIN Episodios e ON s.serie_id = e.serie_id
GROUP BY s.serie_id, s.titulo
ORDER BY Total_Episodios DESC;
```

Concepto Clave

Funciones de Agregación:

- `COUNT()`: Cuenta el número de registros
- `AVG()`: Calcula el promedio
- `SUM()`: Suma valores
- `MAX()`: Encuentra el valor máximo
- `MIN()`: Encuentra el valor mínimo

4.2.4 Actores más jóvenes

```sql
SELECT
    nombre,
    fecha_nacimiento,
    YEAR(CURDATE()) - YEAR(fecha_nacimiento) AS Edad
FROM Actores
ORDER BY fecha_nacimiento DESC
LIMIT 5;
```

4.2.5 Series con más actores

```sql
SELECT
    s.titulo AS Serie,
    COUNT(act.actor_id) AS Numero_Actores
FROM Series s
LEFT JOIN Actuaciones act ON s.serie_id = act.serie_id
GROUP BY s.serie_id, s.titulo
ORDER BY Numero_Actores DESC;
```

5 Proyecto del Día: Base de Datos Pokémon

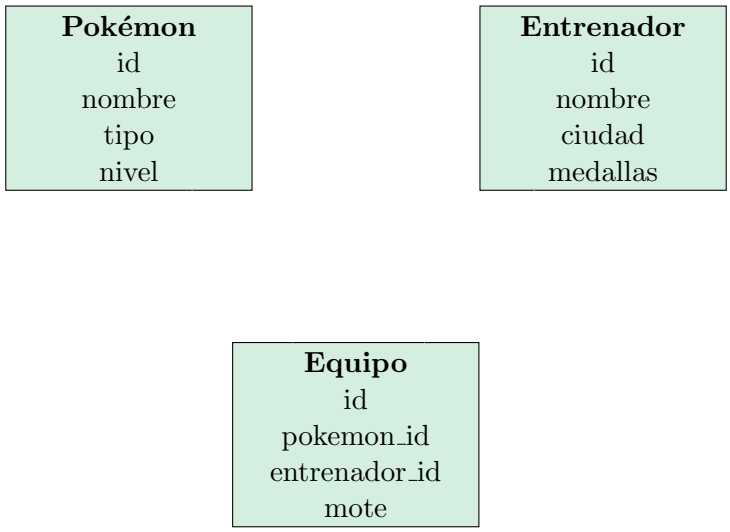
Proyecto del Día

Objetivo del Proyecto

Crearás una base de datos completa llamada `pokemon_db` que contendrá información sobre Pokémon, entrenadores y los equipos que forman. Este proyecto te permitirá practicar:

- Creación de tablas con diferentes tipos de datos
- Implementación de claves primarias y foráneas
- Relaciones muchos a muchos
- Diseño basado en requisitos del mundo real

5.1 Diagrama Entidad-Relación



5.2 Paso 1: Crear la Base de Datos

Instrucciones

Instrucción: Crea una nueva base de datos con un nombre identificable.

```
CREATE DATABASE IF NOT EXISTS pokemon_db;
USE pokemon_db;
```

5.3 Paso 2: Crear las Tablas

5.3.1 Tabla Pokémon

Instrucciones

Esta tabla almacenará los datos de cada Pokémon.
Debe incluir:

- Identificador único (clave primaria)
- Nombre del Pokémon
- Tipo principal (Fuego, Agua, Planta, etc.)
- Tipo secundario (puede ser nulo)
- Región de origen (Kanto, Johto, Sinnoh, etc.)
- Nivel del Pokémon

```sql
CREATE TABLE IF NOT EXISTS Pokemon (
    pokemon_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    tipo_principal VARCHAR(50) NOT NULL,
    tipo_secundario VARCHAR(50),
    region VARCHAR(50),
    nivel INT DEFAULT 1
);
```

Concepto Clave

DEFAULT 1: Esta cláusula establece un valor por defecto. Si no se especifica el nivel al insertar un Pokémon, automáticamente será 1.

5.3.2 Tabla Entrenador

Instrucciones

Representa a los entrenadores Pokémon.
Debe incluir:

- Identificador único (clave primaria)
- Nombre del entrenador
- Ciudad de origen
- Edad
- Número de medallas obtenidas (valor por defecto: 0)

```sql
CREATE TABLE IF NOT EXISTS Entrenador (
    entrenador_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    ciudad VARCHAR(100),
    edad INT,
    medallas INT DEFAULT 0
```

```
);
```

5.3.3 Tabla Equipo

Instrucciones

Esta tabla relaciona entrenadores con Pokémon (relación muchos a muchos).
Debe incluir:

- Identificador único (clave primaria)
- ID del entrenador (clave foránea)
- ID del Pokémon (clave foránea)
- Mote o apodo del Pokémon
- Fecha de captura

```
CREATE TABLE IF NOT EXISTS Equipo (
    equipo_id INT AUTO_INCREMENT PRIMARY KEY,
    entrenador_id INT NOT NULL,
    pokemon_id INT NOT NULL,
    mote VARCHAR(100),
    fecha_captura DATE,
    FOREIGN KEY (entrenador_id) REFERENCES Entrenador(entrenador_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (pokemon_id) REFERENCES Pokemon(pokemon_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Advertencia

ON DELETE CASCADE: Si se elimina un entrenador o un Pokémon, también se eliminarán automáticamente sus registros en la tabla Equipo. Esto mantiene la integridad referencial.

5.4 Paso 3: Insertar Datos de Ejemplo

5.4.1 Insertar Pokémon

Vamos a agregar varios Pokémon de diferentes tipos y regiones:

```
INSERT INTO Pokemon
    (nombre, tipo_principal, tipo_secundario, region, nivel)
VALUES
    ('Pikachu', 'Electrico', NULL, 'Kanto', 25),
    ('Charizard', 'Fuego', 'Volador', 'Kanto', 36),
    ('Blastoise', 'Agua', NULL, 'Kanto', 36),
    ('Venusaur', 'Planta', 'Veneno', 'Kanto', 32),
    ('Gengar', 'Fantasma', 'Veneno', 'Kanto', 40),
    ('Dragonite', 'Dragon', 'Volador', 'Kanto', 55),
    ('Mewtwo', 'Psiquico', NULL, 'Kanto', 70),
    ('Typhlosion', 'Fuego', NULL, 'Johto', 36),
```

```
    ('Feraligatr', 'Agua', NULL, 'Johto', 30),
    ('Meganium', 'Planta', NULL, 'Johto', 32),
    ('Umbreon', 'Siniestro', NULL, 'Johto', 28),
    ('Espeon', 'Psiquico', NULL, 'Johto', 28),
    ('Tyranitar', 'Roca', 'Siniestro', 'Johto', 55),
    ('Lugia', 'Psiquico', 'Volador', 'Johto', 70),
    ('Blaziken', 'Fuego', 'Lucha', 'Hoenn', 36),
    ('Swampert', 'Agua', 'Tierra', 'Hoenn', 36),
    ('Sceptile', 'Planta', NULL, 'Hoenn', 36),
    ('Gardevoir', 'Psiquico', 'Hada', 'Hoenn', 30),
    ('Aggron', 'Acero', 'Roca', 'Hoenn', 42),
    ('Rayquaza', 'Dragon', 'Volador', 'Hoenn', 70),
    ('Infernape', 'Fuego', 'Lucha', 'Sinnoh', 36),
    ('Empoleon', 'Agua', 'Acero', 'Sinnoh', 36),
    ('Torterra', 'Planta', 'Tierra', 'Sinnoh', 32),
    ('Lucario', 'Lucha', 'Acero', 'Sinnoh', 30),
    ('Garchomp', 'Dragon', 'Tierra', 'Sinnoh', 48),
    ('Luxray', 'Electrico', NULL, 'Sinnoh', 30),
    ('Dialga', 'Acero', 'Dragon', 'Sinnoh', 75),
    ('Greninja', 'Agua', 'Siniestro', 'Kalos', 36),
    ('Delphox', 'Fuego', 'Psiquico', 'Kalos', 36),
    ('Sylveon', 'Hada', NULL, 'Kalos', 28);
```

5.4.2 Insertar Entrenadores

Agregamos varios entrenadores de diferentes ciudades:

```
INSERT INTO Entrenador (nombre, ciudad, edad, medallas)
VALUES
    ('Ash Ketchum', 'Pueblo Paleta', 18, 8),
    ('Misty Waterflower', 'Ciudad Celeste', 19, 8),
    ('Brock Harrison', 'Ciudad Plateada', 21, 8),
    ('Gary Oak', 'Pueblo Paleta', 18, 8),
    ('Red', 'Pueblo Paleta', 20, 16),
    ('Blue', 'Pueblo Paleta', 20, 8),
    ('Lance', 'Ciudad Fucsia', 25, 8),
    ('Cynthia', 'Pueblo Arbolado', 27, 8),
    ('Steven Stone', 'Ciudad Oromar', 26, 8),
    ('Leon', 'Pueblo Postwick', 22, 8),
    ('Diantha', 'Ciudad Luminalia', 28, 8),
    ('May Haruka', 'Ciudad Malvalona', 17, 6),
    ('Dawn Hikari', 'Pueblo Arena', 16, 5),
    ('Serena', 'Ciudad Luminalia', 17, 3),
    ('Paul Shinji', 'Ciudad Malvalona', 19, 8);
```

5.4.3 Insertar Equipos

Ahora relacionamos entrenadores con sus Pokémon:

```
INSERT INTO Equipo
    (entrenador_id, pokemon_id, mote, fecha_captura)
VALUES
    -- Equipo de Ash Ketchum (entrenador_id = 1)
    (1, 1, 'Pikachu', '1997-04-01'),
    (1, 2, 'Charizard', '1997-06-15'),
    (1, 5, 'Gengar', '2019-11-10'),
    (1, 21, 'Infernape', '2006-12-15'),
```

```
9       (1, 28, 'Greninja', '2013-10-17'),
10
11      -- Equipo de Misty (entrenador_id = 2)
12      (2, 3, 'Blastoise', '1998-03-20'),
13      (2, 9, 'Feraligatr', '1999-10-14'),
14
15      -- Equipo de Brock (entrenador_id = 3)
16      (3, 19, 'Aggron', '2003-05-08'),
17      (3, 25, 'Garchomp', '2007-09-13'),
18
19      -- Equipo de Gary Oak (entrenador_id = 4)
20      (4, 3, 'Blastoise', '1997-04-10'),
21      (4, 6, 'Dragonite', '1999-01-28'),
22      (4, 13, 'Tyranitar', '1999-10-14'),
23      (4, 11, 'Umbreon', '1999-04-08'),
24
25      -- Equipo de Red (entrenador_id = 5)
26      (5, 1, 'Pika', '1996-02-27'),
27      (5, 2, 'Charizard', '1996-03-15'),
28      (5, 3, 'Blastoise', '1996-04-20'),
29      (5, 4, 'Venusaur', '1996-05-10'),
30      (5, 12, 'Espeon', '1999-11-21'),
31      (5, 7, 'Mewtwo', '1998-09-12'),
32
33      -- Equipo de Lance (entrenador_id = 7)
34      (7, 6, 'Dragonite', '2000-01-15'),
35      (7, 20, 'Rayquaza', '2005-09-16'),
36      (7, 25, 'Garchomp', '2007-09-28'),
37
38      -- Equipo de Cynthia (entrenador_id = 8)
39      (8, 25, 'Garchomp', '2006-09-28'),
40      (8, 24, 'Lucario', '2006-09-30'),
41      (8, 26, 'Luxray', '2007-04-19'),
42      (8, 18, 'Gardevoir', '2003-03-19'),
43
44      -- Equipo de Steven Stone (entrenador_id = 9)
45      (9, 7, 'Mewtwo', '2003-03-19'),
46      (9, 19, 'Aggron', '2003-01-15'),
47      (9, 17, 'Sceptile', '2003-03-19'),
48
49      -- Equipo de Leon (entrenador_id = 10)
50      (10, 2, 'Charizard', '2019-11-15'),
51      (10, 6, 'Dragonite', '2019-11-15'),
52      (10, 27, 'Dialga', '2019-11-15'),
53
54      -- Equipo de May (entrenador_id = 12)
55      (12, 15, 'Blaziken', '2003-03-21'),
56      (12, 17, 'Sceptile', '2004-09-02'),
57      (12, 18, 'Gardevoir', '2004-01-15'),
58
59      -- Equipo de Dawn (entrenador_id = 13)
60      (13, 22, 'Empoleon', '2006-09-28'),
61      (13, 24, 'Lucario', '2007-05-31'),
62      (13, 10, 'Meganium', '2008-02-14'),
63
64      -- Equipo de Serena (entrenador_id = 14)
65      (14, 29, 'Delphox', '2013-10-17'),
66      (14, 30, 'Sylveon', '2014-02-20'),
```

```
67      (14, 28, 'Greninja', '2014-11-13');
```

5.5 Paso 4: Consultas de Verificación

5.5.1 Ver todos los Pokémon capturados

```
SELECT * FROM Pokemon;
```

5.5.2 Ver entrenadores con más medallas

```
SELECT nombre, ciudad, medallas
FROM Entrenador
ORDER BY medallas DESC;
```

5.5.3 Ver equipos completos (JOIN)

Ejercicio Práctico

Consulta que muestra:

- Nombre del entrenador
- Nombre del Pokémon
- Mote del Pokémon
- Tipo principal
- Nivel

```
SELECT
    e.nombre AS Entrenador,
    p.nombre AS Pokemon,
    eq.mote AS Apodo,
    p.tipo_principal AS Tipo,
    p.nivel AS Nivel,
    eq.fecha_captura AS Fecha_Captura
FROM Equipo eq
INNER JOIN Entrenador e ON eq.entrenador_id = e.entrenador_id
INNER JOIN Pokemon p ON eq.pokemon_id = p.pokemon_id
ORDER BY e.nombre, p.nivel DESC;
```

5.5.4 Contar Pokémon por entrenador

```
SELECT
    e.nombre AS Entrenador,
    COUNT(eq.pokemon_id) AS Cantidad_Pokemon,
    e.medallas AS Medallas
FROM Entrenador e
LEFT JOIN Equipo eq ON e.entrenador_id = eq.entrenador_id
GROUP BY e.entrenador_id, e.nombre, e.medallas
ORDER BY Cantidad_Pokemon DESC;
```

5.5.5 Pokémon por tipo

```sql
SELECT
    tipo_principal,
    COUNT(*) AS Total
FROM Pokemon
GROUP BY tipo_principal
ORDER BY Total DESC;
```

5.5.6 Pokémon de nivel más alto por región

```sql
SELECT
    region,
    nombre,
    nivel,
    tipo_principal
FROM Pokemon
WHERE (region, nivel) IN (
    SELECT region, MAX(nivel)
    FROM Pokemon
    GROUP BY region
)
ORDER BY nivel DESC, region;
```

5.5.7 Entrenadores y su Pokémon más poderoso

```sql
SELECT
    e.nombre AS Entrenador,
    p.nombre AS Pokemon_Mas_Fuerte,
    p.nivel AS Nivel,
    p.tipo_principal AS Tipo
FROM Entrenador e
INNER JOIN Equipo eq ON e.entrenador_id = eq.entrenador_id
INNER JOIN Pokemon p ON eq.pokemon_id = p.pokemon_id
WHERE p.nivel = (
    SELECT MAX(p2.nivel)
    FROM Equipo eq2
    INNER JOIN Pokemon p2 ON eq2.pokemon_id = p2.pokemon_id
    WHERE eq2.entrenador_id = e.entrenador_id
)
ORDER BY p.nivel DESC;
```

5.6 Paso 5: Desafío Extra

Ejercicio Práctico

Desafío 1: Agregar Columnas
Agrega las siguientes columnas:
En la tabla Pokemon:

```sql
ALTER TABLE Pokemon
ADD COLUMN puntos_experiencia INT DEFAULT 0;

ALTER TABLE Pokemon
ADD COLUMN es_legendario BOOLEAN DEFAULT FALSE;
```

En la tabla Entrenador:

```sql
ALTER TABLE Entrenador
ADD COLUMN torneos_ganados INT DEFAULT 0;

ALTER TABLE Entrenador
ADD COLUMN fecha_registro DATE;
```

Ejercicio Práctico

Desafío 2: Actualizar Datos
Marca algunos Pokémon como legendarios:

```sql
UPDATE Pokemon
SET es_legendario = TRUE
WHERE nombre IN ('Mewtwo', 'Lugia', 'Rayquaza', 'Dialga');
```

Actualiza los torneos ganados de algunos entrenadores:

```sql
UPDATE Entrenador
SET torneos_ganados = 3
WHERE nombre = 'Red';

UPDATE Entrenador
SET torneos_ganados = 2
WHERE nombre IN ('Cynthia', 'Leon');
```

Ejercicio Práctico

Desafío 3: Crear una Vista
Crea una vista que muestre información completa de cada entrenador con su equipo:

```sql
CREATE VIEW Vista_Equipos_Completos AS
SELECT
    e.nombre AS Entrenador,
    e.ciudad AS Ciudad,
    e.medallas AS Medallas,
    e.torneos_ganados AS Torneos,
    p.nombre AS Pokemon,
    p.tipo_principal AS Tipo_Principal,
    p.tipo_secundario AS Tipo_Secundario,
    p.nivel AS Nivel,
```

```
    p.region AS Region,
    p.es_legendario AS Es_Legendario,
    eq.mote AS Apodo,
    eq.fecha_captura AS Fecha_Captura
FROM Entrenador e
INNER JOIN Equipo eq ON e.entrenador_id = eq.entrenador_id
INNER JOIN Pokemon p ON eq.pokemon_id = p.pokemon_id;
```

Ahora puedes consultar la vista fácilmente:

```
-- Ver todos los datos
SELECT * FROM Vista_Equipos_Completos;

-- Ver solo entrenadores con Pok mon legendarios
SELECT DISTINCT Entrenador, Pokemon, Tipo_Principal
FROM Vista_Equipos_Completos
WHERE Es_Legendario = TRUE;
```

Concepto Clave

¿Qué es una Vista (VIEW)?
Una vista es una tabla virtual basada en el resultado de una consulta SQL. Las vistas:

- No almacenan datos físicamente
- Se actualizan automáticamente cuando cambian las tablas base
- Simplifican consultas complejas
- Mejoran la seguridad al ocultar columnas sensibles
- Facilitan el acceso a datos frecuentemente consultados

6 Ejercicios Adicionales

Ejercicio Práctico

Ejercicio 1: Análisis de Tipos
Crea una consulta que muestre cuántos Pokémon de cada tipo tiene cada entrenador.

Ejercicio 2: Ranking de Entrenadores
Crea un ranking de entrenadores basado en:

- Número total de medallas
- Torneos ganados
- Número de Pokémon legendarios en su equipo

Ejercicio 3: Pokémon sin Capturar
Encuentra todos los Pokémon que no han sido capturados por ningún entrenador.

Ejercicio 4: Estadísticas por Región
Crea un reporte que muestre para cada región:

- Número total de Pokémon
- Nivel promedio
- Tipo más común
- Número de legendarios

6.1 Soluciones a los Ejercicios

6.1.1 Solución Ejercicio 1

```
SELECT
    e.nombre AS Entrenador,
    p.tipo_principal AS Tipo,
    COUNT(*) AS Cantidad
FROM Entrenador e
INNER JOIN Equipo eq ON e.entrenador_id = eq.entrenador_id
INNER JOIN Pokemon p ON eq.pokemon_id = p.pokemon_id
GROUP BY e.entrenador_id, e.nombre, p.tipo_principal
ORDER BY e.nombre, Cantidad DESC;
```

6.1.2 Solución Ejercicio 2

```
SELECT
    e.nombre AS Entrenador,
    e.medallas AS Medallas,
    e.torneos_ganados AS Torneos,
    COUNT(CASE WHEN p.es_legendario = TRUE THEN 1 END) AS Legendarios
        ,
    (e.medallas * 10 + e.torneos_ganados * 50 +
```

```
      COUNT(CASE WHEN p.es_legendario = TRUE THEN 1 END) * 100) AS
         Puntuacion
FROM Entrenador e
LEFT JOIN Equipo eq ON e.entrenador_id = eq.entrenador_id
LEFT JOIN Pokemon p ON eq.pokemon_id = p.pokemon_id
GROUP BY e.entrenador_id, e.nombre, e.medallas, e.torneos_ganados
ORDER BY Puntuacion DESC;
```

6.1.3 Solución Ejercicio 3

```
SELECT
    p.nombre AS Pokemon,
    p.tipo_principal AS Tipo,
    p.region AS Region,
    p.nivel AS Nivel
FROM Pokemon p
LEFT JOIN Equipo eq ON p.pokemon_id = eq.pokemon_id
WHERE eq.equipo_id IS NULL
ORDER BY p.nivel DESC;
```

6.1.4 Solución Ejercicio 4

```
SELECT
    region AS Region,
    COUNT(*) AS Total_Pokemon,
    ROUND(AVG(nivel), 2) AS Nivel_Promedio,
    (SELECT tipo_principal
     FROM Pokemon p2
     WHERE p2.region = p1.region
     GROUP BY tipo_principal
     ORDER BY COUNT(*) DESC
     LIMIT 1) AS Tipo_Mas_Comun,
    SUM(CASE WHEN es_legendario = TRUE THEN 1 ELSE 0 END) AS
       Legendarios
FROM Pokemon p1
GROUP BY region
ORDER BY Total_Pokemon DESC;
```

7 Conceptos Avanzados

7.1 Índices para Mejorar el Rendimiento

Concepto Clave

Los **índices** mejoran la velocidad de las consultas al crear estructuras de datos optimizadas para búsquedas rápidas.

```sql
-- Crear indices en columnas frecuentemente consultadas
CREATE INDEX idx_pokemon_tipo ON Pokemon(tipo_principal);
CREATE INDEX idx_pokemon_region ON Pokemon(region);
CREATE INDEX idx_entrenador_ciudad ON Entrenador(ciudad);
CREATE INDEX idx_equipo_entrenador ON Equipo(entrenador_id);
CREATE INDEX idx_equipo_pokemon ON Equipo(pokemon_id);
```

7.2 Transacciones

Concepto Clave

Las **transacciones** agrupan múltiples operaciones en una unidad atómica que se ejecuta completamente o no se ejecuta en absoluto.

```sql
-- Ejemplo: Transferir un Pokemon entre entrenadores
START TRANSACTION;

-- Eliminar Pokemon del equipo del entrenador 1
DELETE FROM Equipo
WHERE entrenador_id = 1 AND pokemon_id = 5;

-- Agregar Pokemon al equipo del entrenador 2
INSERT INTO Equipo (entrenador_id, pokemon_id, mote, fecha_captura)
VALUES (2, 5, 'Gengar', CURDATE());

-- Si todo esta bien, confirmar los cambios
COMMIT;

-- Si algo sale mal, revertir los cambios
-- ROLLBACK;
```

7.3 Procedimientos Almacenados

Concepto Clave

Los **procedimientos almacenados** son conjuntos de instrucciones SQL que se almacenan en la base de datos y pueden ejecutarse repetidamente.

```sql
DELIMITER //

CREATE PROCEDURE CapturarPokemon(
    IN p_entrenador_id INT,
    IN p_pokemon_id INT,
```

```
    IN p_mote VARCHAR(100)
)
BEGIN
    INSERT INTO Equipo (entrenador_id, pokemon_id, mote,
        fecha_captura)
    VALUES (p_entrenador_id, p_pokemon_id, p_mote, CURDATE());

    SELECT CONCAT('Pokemon capturado exitosamente por ',
                  (SELECT nombre FROM Entrenador WHERE entrenador_id
                      = p_entrenador_id),
                  '!') AS Mensaje;
END //

DELIMITER ;

-- Usar el procedimiento
CALL CapturarPokemon(1, 15, 'Blaze');
```

8 Buenas Prácticas de Diseño

8.1 Normalización de Bases de Datos

Concepto Clave

Normalización es el proceso de organizar datos para reducir redundancia y mejorar la integridad.
Formas Normales Principales:

1. **Primera Forma Normal (1FN):** Eliminar grupos repetidos
2. **Segunda Forma Normal (2FN):** Eliminar dependencias parciales
3. **Tercera Forma Normal (3FN):** Eliminar dependencias transitivas

8.2 Naming Conventions

- ✔ Usa nombres descriptivos y consistentes
- ✔ Prefieres snake_case o camelCase consistentemente
- ✔ Tablas en plural o singular (elige uno)
- ✔ Claves primarias: `tabla_id`
- ✔ Claves foráneas: `tabla_referenciada_id`
- ✔ Índices: `idx_tabla_columna`

8.3 Documentación

```
-- Siempre documenta tu codigo con comentarios claros

-- Tabla: Pokemon
-- Descripcion: Almacena informacion de todos los Pokemon disponibles
-- Relaciones: 1:N con Equipo

-- Tabla: Equipo
-- Descripcion: Tabla intermedia que relaciona Entrenadores con
    Pokemon
-- Tipo: Tabla de union para relacion N:N
```

9 Resumen de la Clase

Concepto Clave

Conceptos Clave Aprendidos:

1. **Bases de datos complejas:** Creación de sistemas con múltiples tablas relacionadas
2. **Relaciones N:N:** Implementación mediante tablas intermedias
3. **Tipos de datos avanzados:** DECIMAL, TEXT, DATE, BOOLEAN
4. **Consultas JOIN:** Unir datos de múltiples tablas
5. **Funciones de agregación:** COUNT, AVG, SUM, MAX, MIN
6. **Vistas:** Simplificar consultas complejas
7. **Índices:** Mejorar el rendimiento
8. **Integridad referencial:** CASCADE en claves foráneas

10 Checklist de Verificación

- [ ] He creado la base de datos NetflixDB completa
- [ ] He insertado todas las series y actores
- [ ] Entiendo las relaciones uno a muchos
- [ ] Entiendo las relaciones muchos a muchos
- [ ] He creado la base de datos pokemon_db
- [ ] He insertado Pokémon, entrenadores y equipos
- [ ] Puedo realizar consultas con JOIN
- [ ] Sé usar funciones de agregación
- [ ] He creado al menos una vista
- [ ] He completado los ejercicios adicionales

11 Recursos Adicionales

11.1 Para Seguir Practicando

- Expande la base de datos con más tablas (Gimnasios, Movimientos, Objetos)
- Crea consultas más complejas con subconsultas
- Implementa triggers para automatizar acciones
- Diseña tu propia base de datos desde cero

11.2 Próximos Pasos

En la siguiente clase aprenderás:

- Subconsultas avanzadas
- Funciones de ventana (Window Functions)
- Triggers y eventos automatizados
- Optimización de consultas
- Stored procedures avanzados

¡Excelente Trabajo!

Has completado la Clase 3 de SQL 1025.
Ahora puedes diseñar y crear bases de datos
relacionales complejas con múltiples tablas.

Continúa practicando con tus propios proyectos.