Universidad Nacional Autónoma de México

Facultad de Ciencias, 2026-I

Fundamentos de Bases de Datos

README:

AMIGUIBIS

Integrantes:

Elizalde Maza Jesús Eduardo, 321031686
Gòmez Calva Carlos Manuel, 321281678
Lugo Díaz Ordaz Gretel Alexandra, 321115128
Peredo López Citlalli Abigail, 321161022
Santana de la Rosa Monica Guadalupe, 321023663

03 Diciembre 2025
México

Índice general

Capítulo 1

Objetivos

El presente documento detalla la arquitectura, diseño e implementación de la base de datos relacional desarrollada para el sistema **SGTP** (Sistema de Gestión de Torneos Pokémon). Este sistema tiene como objetivo centralizar la administración de eventos masivos, el control de participantes, la logística de personal (vendedores, limpiadores, registradores) y la gestión detallada de las competencias (torneos de peleas, captura de shinys y distancia).

La solución implementada en **PostgreSQL** garantiza la integridad de los datos, optimiza las consultas complejas mediante un diseño normalizado y asegura la escalabilidad para futuras ediciones de los eventos.

1.1. Objetivo del SGTP

Este sistema busca centralizar la administración integral de eventos masivos, abarcando:

- Control de participantes.
- Logística de personal (registradores, vendedores, limpiadores, etc.).
- Gestión detallada de las diversas competencias (torneos de peleas, captura de shinys y distancia).

1.2. Solución Técnica

La solución se implementó utilizando **PostgreSQL**, garantizando la integridad de los datos. El diseño, basado en la normalización, optimiza las consultas complejas y asegura la escalabilidad del sistema para futuras ediciones de los eventos.

Capítulo 2

Análisis de Requerimientos

2.1. Requerimientos Candidatos

- Registro de participantes con datos personales y cuentas de Pokémon Go.
- Registro de los tres tipos de torneos: peleas, distancia recorrida y captura de shinys.
- Registro del personal organizador (cuidadores, limpiadores, vendedores y personal de registro).
- Registro de visitantes.
- Registro de premios y ganadores.
- Control de ediciones del torneo.
- Registro de ventas de alimentos y cálculo de ganancias.
- Cálculo de pagos a personal organizador.

2.2. Comprensión del Contexto del Sistema

- Participantes → inscripción, torneos, Pokémon utilizados, distancias, capturas.
- Organizador (administrativo) → controlar torneos, registrar personal, calcular pagos y ganancias.
- Personal de registro → registrar participantes y generar costo por registro.
- Vendedores → registrar ventas, productos y calcular ganancias individuales.
- Visitantes → registro de entrada/salida, métodos de pago.
- Sistema → centraliza toda la información para que el torneo se realice correctamente.

2.3. Requerimientos Funcionales

2.3.1. Participantes

El sistema debe permitir registrar:

- Nombre completo.
- Fecha de nacimiento.
- Edad.
- Sexo.
- Teléfonos y correos.
- Número de cuenta UNAM.
- Facultad y carrera.

Cada participante debe registrar al menos una cuenta Pokémon Go:

- Código de entrenador.
- Nombre de usuario.
- Nivel.
- Equipo (Sabiduría, Instinto o Valor).

2.3.2. Torneos

Torneo de Peleas

- Pokémon participantes (máximo 6 por usuario).
- Enfrentamientos efectuados.
- Pokémon usados en cada pelea.
- Ganador del enfrentamiento.

Torneo de Distancia Recorrida

- Distancia inicial.
- Distancias registradas en CU, Rectoría y Universum.
- Hora de llegada a cada locación.
- Validación de que el participante visitó al menos una locación.

Torneo de Captura de Shinys

- Nombre o apodo.
- Especie.
- Tipo.
- CP.
- Peso.
- Sexo.
- Shiny o no shiny.
- Fecha y hora de captura.

Restricción de participación

- Participantes del torneo de peleas no pueden participar en otros torneos.
- Participantes de distancia y shinys pueden participar simultáneamente.

2.4. Personal Organizador

2.4.1. Cuidadores y Limpiadores

Registrar:

- Datos personales (nombre, nacimiento, edad, sexo, teléfonos, correos).
- Dirección completa.
- Horario (9–3 o 3–9).
- Locación asignada.
- Salario.

2.4.2. Vendedores

Además de datos personales y dirección, registrar:

- Datos de alimentos (tipo, nombre, perecedero, fecha de caducidad).
- Precio sin IVA y precio con IVA.
- Ubicación del puesto.
- Ganancia del 25 % de ventas totales.

2.4.3. Personal de registro

Registrar:

- Datos personales.
- Número de personas registradas.
- Pago total (250 por registro + 50 extra por persona).
- Horario especial (7–9 am).

2.5. Visitantes

El sistema debe registrar:

- Nombre completo.
- Edad.
- Fecha de nacimiento.
- Sexo.
- Hora de ingreso y salida.
- Si compran alimentos: método de pago (efectivo, PayPal, tarjeta).

2.6. Premios y Resultados

El sistema debe registrar:

- Ganadores.
- Pago de 5,000 pesos.
- Trofeo y mercancía.
- Resultados por torneo.

2.7. Control de Ediciones

Registrar:

- Número de edición.
- Fecha del evento.
- Dos eventos por año.

2.8. Requerimientos No Funcionales

2.8.1. Asociados

- Validación estricta de datos UNAM.
- Validación de máximo 6 Pokémon en peleas.
- Validación de locaciones visitadas en distancia.
- Validación temporal de capturas dentro del evento.

2.8.2. No Asociados

- Seguridad y protección de datos personales.
- Alta disponibilidad del sistema durante el torneo.
- Escalabilidad para futuras ediciones.
- Integridad y consistencia de datos.
- Rendimiento adecuado para consultas.

Capítulo 3

Modelo Conceptual

3.1. Identificadores

- Persona → IdPersona
- Participante → Número_cuenta
- Organizador → IdOrganizador (por herencia)
- Visitante → (hereda de Persona, no identificador propio)
- Vendedor → (hereda de Persona, no identificador propio)
- Cuenta_pokemon → Código_entrenador
- Pokémon → IdPokemon
- Evento → IdEvento
- Torneo → IdTorneo
- TorneoPeleas / TorneoDistancia / TorneoShinys → (heredan IdTorneo)
- Enfrentamiento → IdEnfrentamiento
- Premio → IdPremio
- Alimento → IdAlimento

3.2. Relaciones (Cardinalidad y Participación)

3.2.1. Pertenece (Pokémon – Participante)

- Varios Pokémon pertenecen a un participante.
- A un participante le pertenecen muchos Pokémon.
- Cardinalidad: 1:N
- Participación: Parcial de ambos lados.

3.2.2. Tener (Cuenta_pokemon – Participante)

- Una cuenta_pokemon puede tener un participante.
- A cada participante le pertenecen muchas cuentas_pokemon.
- Cardinalidad: 1:N
- Participación: Total por ambos lados.

3.2.3. Participar (Participante – Torneo)

- Un participante puede inscribirse en muchos torneos.
- Un torneo puede tener muchos participantes.
- Cardinalidad: N:M
- Participación: Parcial en ambos lados.

3.2.4. Desafía / Es-desafiado (Participante – Enfrentamiento)

- Un participante puede desafiar y ser desafiado en muchos enfrentamientos.
- Varios enfrentamientos pueden involucrar a un participante.
- Cardinalidad: 1:N
- Participación: Total por ambos lados.

3.2.5. Genera (TorneoPeleas – Enfrentamiento)

- Un TorneoPeleas puede generar muchos enfrentamientos.
- Cada enfrentamiento se genera en un solo torneo.
- Cardinalidad: 1:N
- Participación: Total por ambos lados.

3.2.6. Gana (Participante – Premio)

- Un participante puede ganar muchos premios.
- Un premio solo puede ser asignado a un participante.
- Cardinalidad: 1:N
- Participación: Parcial en ambos lados.

3.2.7. Compra (Persona – Alimento)

- Una persona puede comprar muchos alimentos.
- Un alimento puede ser comprado por muchos participantes.
- Cardinalidad: N:M
- Atributo: MétodoPago.
- Participación: Total por ambos lados.

3.2.8. Registrar (Persona – Registrador)

- Muchas personas pueden registrarse en un registro.
- En un registro solo puede estar registrada una vez esa persona.
- Cardinalidad: 1:N
- Participación: Total por ambos lados.

3.2.9. Asistir (Persona – Evento)

- Una persona puede asistir a muchos eventos.
- Un evento puede tener muchas personas asistentes.
- Cardinalidad: N:M
- Participación: Total por ambos lados.

3.3. Consideraciones

- Los torneos entre 2 personas solo pueden tener un máximo de 6 Pokémon.
- Las peleas en el Torneo de Peleas se realizan siempre entre dos participantes.
- Si un participante se inscribe en el Torneo de Peleas, no podrá inscribirse en otro torneo (ni de Distancia ni de Shinys).
- Todo Evento debe estar registrado por al menos un Organizador.
- Un Participante puede asistir a eventos sin necesidad de estar inscrito en un torneo.
- Todo Pokémon debe pertenecer obligatoriamente a una Cuenta_pokemon.
- Cada Cuenta_pokemon puede tener múltiples Pokémon, pero un Pokémon no puede pertenecer a más de una cuenta.

- Los Premios están asociados al resultado de un enfrentamiento o torneo, pero siempre se asignan a un solo participante ganador.
- Un Enfrentamiento siempre ocurre dentro de un Torneo y genera exactamente un ganador.
- Un Participante puede comprar múltiples Alimentos, y un mismo alimento puede ser adquirido por distintos participantes.
- Cada compra debe registrar el método de pago (efectivo, tarjeta, etc.).
- El atributo Edad de Persona se calcula a partir de la Fecha_nacimiento.
- Las Ganancias de un Evento se calculan en función de inscripciones, ventas o compras, y no se almacenan directamente.

Capítulo 4

Modelo Lógico

4.1. Traducción

4.1.1. Teléfono

- `Telefono(IDPERSONA: int, telefono: char(10))`
- PK: telefono
- PK: IDPERSONA
- Asociado a: Participante (PK IDPERSONA), Visitante (PK IDPERSONA), Registrador (PK IDPERSONA), Limpiador/Cuidador (PK IDPERSONA), Vendedor (PK IDPERSONA).

4.1.2. Participante

- `Participante(NUMERO_CUENTA: int CK, IDPERSONA: int PK, sexo: varchar(30), fecha_nacimiento: date, edad_calculada: int, nombre: varchar(50), apellido_paterno: varchar(50), apellido_materno: varchar(50), correo: varchar(50), facultad: varchar(50), carrera: varchar(50))`
- PK: NUMERO_CUENTA

4.1.3. Visitante

- `Visitante(IDPERSONA PK, sexo: varchar(20), fecha_nacimiento: date, edad_calculada: int, nombre: varchar(50), apellido_paterno: varchar(50), apellido_materno: varchar(50), horaentrada: time, horasalida: time)`
- PK: IDPERSONA

4.1.4. Correo

- `Correo(IDORGANIZADOR: int, correo: varchar(50))`
- PK: (IDORGANIZADOR, correo)

- FK: IDORGANIZADOR → Organizador(IDORGANIZADOR)

4.1.5. Vendedor

- `Vendedor(IDORGANIZADOR: int PK, IDPERSONA: int, sexo: varchar(20), fecha_nacimiento: date, edad_calculada: int, nombre: varchar(50), apellido_paterno: varchar(50), apellido_materno: varchar(50), locacion: varchar(50), pago_calculado: int)`
- PK: IDORGANIZADOR
- FK: IDORGANIZADOR → Organizador(IDORGANIZADOR)

4.1.6. Limpiador

- `Limpiador(IDORGANIZADOR: int PK, IDPERSONA: int, sexo: varchar(20), fecha_nacimiento: date, edad_calculada: int, nombre: varchar(50), apellido_paterno: varchar(50), apellido_materno: varchar(50), locacion: varchar(50), pago_calculado: int, horario: time, puesto: varchar(50))`
- PK: IDORGANIZADOR
- FK: IDORGANIZADOR → Organizador(IDORGANIZADOR)

4.1.7. Registrador

- `Registrador(IDORGANIZADOR: int PK, IDPERSONA: int, sexo: varchar(20), fecha_nacimiento: date, edad_calculada: int, nombre: varchar(50), apellido_paterno: varchar(50), apellido_materno: varchar(50), pago_calculado: int)`
- PK: IDORGANIZADOR
- FK: IDORGANIZADOR → Organizador(IDORGANIZADOR)

4.1.8. Torneos

- `TorneoPeleas(IDTORNEO: int PK, fechainicio: date, fechafinal: date, reglas: varchar(100))`
- `TorneoDistancia(IDTORNEO: int PK, fechainicio: date, fechafinal: date, reglas: varchar(100), locacionpartida: varchar(50))`
- `PuntosLocacion(IDTORNEO: int, puntoslocacion: int)` PK: (IDTORNEO, puntoslocacion)
- `TorneoShinys(IDTORNEO: int PK, fechainicio: date, fechafinal: date, reglas: varchar(100))`

4.1.9. Enfrentamiento

- `Enfrentamiento(IDTORNEO: int, IDENTFRENTAMIENTO: int, numero_Cuenta: char(9), ganador: varchar(40), resultado: varchar(30), participante_desafiado_I participante_desafiante_IDPERSONA)`
- PK: (IDTORNEO, IDENTFRENTAMIENTO)
- FKs:
  - IDTORNEO → TorneoPeleas/TorneoDistancia/TorneoShinys(IDTORNEO)
  - participante_desafiado_IDPERSONA → Participante(IDPERSONA)
  - participante_desafiante_IDPERSONA → Participante(IDPERSONA)

4.1.10. Premio

- `Premio(IDPREMIO: int, dinero: money, mercancia: varchar(100), trofeo: varchar(60))`
- PK: IDPREMIO

4.1.11. Pokémon

- `Pokemon(IDPOKEMON: int, sexo: varchar(20), especie: varchar(50), peso: numeric, shiny: boolean, CP: char(5), nombre: varchar(100), horaCaptura: time, IDPERSONA: int)`
- PK: IDPOKEMON
- FK: IDPERSONA → Participante(IDPERSONA)

4.1.12. Tipo

- `Tipo(IDPOKEMON: int, Tipo: varchar(40))`
- PK compuesta: (IDPOKEMON, Tipo)
- FK: IDPOKEMON → Pokemon(IDPOKEMON)

4.1.13. Cuenta_pokemon

- `Cuenta_pokemon(IDPERSONA: int, CODIGOENTRENADOR: int, nombre_usuario: varchar(50), nivel_entrenador: varchar(50), equipo: varchar(100))`
- PK: (IDPERSONA, CODIGOENTRENADOR)

4.1.14. Evento

- `Evento(IDEVENTO: int, ganancias_calculadas: int, numedicion: int, fecha: date)`
- PK: IDEVENTO

4.1.15. Participar

- `Participar(IDPERSONA: int, IDTORNEO: int)`
- PK: (IDPERSONA, IDTORNEO)
- FKs:
  - IDPERSONA → Participante(IDPERSONA)
  - IDTORNEO → TorneoPeleas/TorneoDistancia/TorneoShinys(IDTORNEO)

4.1.16. Alimento

- `Alimento(IDALIMENTO: int, Nombre: varchar(80), Tipo: varchar(50), Perecedero: boolean, Fecha: date, PrecioVenta: int)`
- PK: IDALIMENTO

4.1.17. Vender

- `Vender(IDALIMENTO: int, IDPERSONA: int)`
- PK compuesta: (IDALIMENTO, IDPERSONA)
- FKs:
  - IDALIMENTO → Alimento(IDALIMENTO)
  - IDPERSONA → Vendedor(IDPERSONA)

4.1.18. Comprar

- `Comprar(IDALIMENTO: int, IDPERSONA: int, MetodoPago: varchar(50))`
- PK compuesta: (IDALIMENTO, IDPERSONA)
- FKs:
  - IDALIMENTO → Alimento(IDALIMENTO)

Capítulo 5

Restricciones de Dominio

5.1. Participante

Las columnas de la tabla `Participante` tienen las siguientes restricciones:

- IdPersona: debe ser mayor que 0
- Ciudad: no debe estar vacía
- Fecha_nacimiento: no debe ser nula
- Apellido_materno: no debe estar vacío
- Apellido_paterno: no debe estar vacío
- Nombre: no debe estar vacío
- Sexo: no debe estar vacío
- Calle: no debe estar vacía
- Colonia: no debe estar vacía
- CP: debe tener longitud igual a 5 caracteres

5.2. Correo

- IdPersona: debe ser mayor que 0
- Correo: no debe estar vacío

5.3. Evento

- IdEvento: debe ser mayor que 0
- NumEdicion: debe ser mayor que 0
- Fecha: no debe ser nula

5.4. Vendedor

- IdPersona: debe ser mayor que 0
- IdOrganizador: debe ser mayor que 0
- IdEvento: debe ser mayor que 0
- Ciudad: no debe estar vacía
- Fecha_nacimiento: no debe ser nula
- Apellido_materno: no debe estar vacío
- Apellido_paterno: no debe estar vacío
- Nombre: no debe estar vacío
- Sexo: no debe estar vacío
- Calle: no debe estar vacía
- Colonia: no debe estar vacía
- CP: debe tener longitud igual a 5 caracteres
- Conteo: debe ser mayor o igual a 0

5.5. Vender

- IdAlimento: debe ser mayor que 0
- IdPersona: debe ser mayor que 0

5.6. Alimento

- IdAlimento: debe ser mayor que 0
- IdPersona: debe ser mayor que 0
- Nombre: no debe estar vacío
- Tipo: no debe estar vacío
- Fecha: no debe ser nula
- PrecioVenta: debe ser mayor o igual a 0

5.7. Comprar

- IdAlimento: debe ser mayor que 0
- IdPersona: debe ser mayor que 0
- MetodoPago: no debe estar vacío

5.8. Teléfono

- IdPersona: debe ser mayor que 0
- Teléfono: debe tener longitud igual a 10 caracteres

5.9. CuentaPokemon

- IdPersona: debe ser mayor que 0
- Cuenta: no debe estar vacía

5.10. Pokémon

- IdPokemon: debe ser mayor que 0
- IdPersona: debe ser mayor que 0
- Nombre: no debe estar vacío
- Nivel: debe ser mayor o igual a 1

5.11. Tipo

- IdPokemon: debe ser mayor que 0
- Tipo: no debe estar vacío

5.12. Participar

- IdTorneo: debe ser mayor que 0
- IdPersona: debe ser mayor que 0

5.13. Ganar

- IdPremio: debe ser mayor que 0
- IdPersona: debe ser mayor que 0
- IdTorneo: debe ser mayor que 0
- IdEnfrentamiento: debe ser mayor que 0

5.14. Enfrentamiento

- IdTorneo: debe ser mayor que 0
- IdEnfrentamiento: debe ser mayor que 0

5.15. Registrador

- IdOrganizador: debe ser mayor que 0
- IdEvento: debe ser mayor que 0

5.16. LimpiadorCuidador

- IdOrganizador: debe ser mayor que 0
- IdEvento: debe ser mayor que 0

5.17. Visitante

- IdEvento: debe ser mayor que 0
- IdPersona: debe ser mayor que 0

5.18. Premio

- IdPremio: debe ser mayor que 0
- Nombre: no debe estar vacío

Capítulo 6

Políticas de Mantenimiento de Llaves Foráneas

Una política de mantenimiento de llaves foráneas define las reglas para preservar la integridad referencial en una base de datos relacional. Las llaves foráneas conectan tablas entre sí, evitando datos inconsistentes y asegurando que toda referencia apunte a un registro válido.

Estas políticas abarcan cuatro áreas principales:

6.1. Comportamiento ante Eliminaciones y Actualizaciones (ON DELETE / ON UPDATE)

Define qué sucede en la tabla hija cuando se modifica o elimina un registro de la tabla padre. Las acciones más comunes son:

- **CASCADE**: Propaga la eliminación o actualización a los registros relacionados.
- **SET NULL**: Establece el valor foráneo en NULL.
- **SET DEFAULT**: Asigna un valor por defecto.
- **RESTRICT / NO ACTION**: Impide la operación si existen dependencias.

Objetivo: evitar referencias inválidas. **Funcionamiento**: se especifica en la definición de la llave foránea.

6.2. Validación de Existencia de Valores Referenciados

- El motor de la base de datos verifica automáticamente que los valores insertados en las llaves foráneas existan en la tabla referenciada.
- **Objetivo**: impedir valores huérfanos.

- **Funcionamiento**: si se inserta un valor no existente, el DBMS genera un error.
- Puede complementarse con triggers si se requieren validaciones más elaboradas.

6.3. Revisión y Mantenimiento Periódico de Relaciones

Consiste en verificar regularmente que las relaciones sigan siendo válidas y eficientes. Incluye herramientas como:

- DBCC CHECKCONSTRAINTS
- DBCC CHECKDB
- Revisión de índices y búsqueda de registros huérfanos.

Objetivo: detectar inconsistencias y mantener buen rendimiento.

6.4. Normas para Crear, Modificar y Eliminar Llaves Foráneas

- **Creación**: `ALTER TABLE ... ADD CONSTRAINT ... FOREIGN KEY`
- **Modificación**: se elimina y se crea de nuevo.
- **Eliminación**: `ALTER TABLE ... DROP CONSTRAINT`

Objetivo: mantener un control claro y consistente de la estructura relacional.

6.5. Ventajas y Desventajas de Cada Política

6.5.1. ON DELETE / ON UPDATE

Ventajas:

- CASCADE automatiza cambios.
- RESTRICT evita pérdidas accidentales.

Desventajas:

- CASCADE puede borrar muchos datos sin aviso.
- SET NULL puede generar registros incompletos.

6.5.2. Validación de existencia

Ventajas:

- Garantiza coherencia entre tablas.

Desventajas:

- Puede complicar migraciones y cargas masivas.

6.5.3. Revisión periódica

Ventajas:

- Detecta errores y optimiza rendimiento.

Desventajas:

- Requiere tiempo y recursos.

6.5.4. Gestión estructural (creación, modificación, eliminación)

Ventajas:

- Mantiene claridad en el modelo.

Desventajas:

- Modificar llaves foráneas implica eliminarlas y recrearlas.

6.6. Política Seleccionada: CASCADE

La política elegida para el esquema es usar **CASCADE** en eliminaciones y actualizaciones, debido a:

- Automatización de la integridad referencial, evitando escribir código adicional para limpieza y mantenimiento.
- Prevención automática de registros huérfanos.
- Simplificación de la lógica de negocio, ya que las entidades dependientes no tienen sentido sin su entidad padre.
- Coherencia del modelo, pues todas las dependencias se actualizan o eliminan en una sola transacción.

Capítulo 7

Poblamiento de la Base de Datos

Para la generación de datos sintéticos utilizados en esta práctica se empleó la herramienta en línea **Mockaroo**, la cual permite crear conjuntos de datos personalizados en formatos como CSV, SQL, JSON, entre otros. Mockaroo fue seleccionada debido a su facilidad de uso, flexibilidad para definir tipos de datos y su capacidad para generar grandes volúmenes de información que cumplen con las restricciones del modelo relacional.

7.1. Proceso de Generación de Datos

7.1.1. Definición de Campos

Para cada tabla del modelo se definieron los atributos correspondientes en Mockaroo, respetando:

- Los nombres exactos establecidos en el archivo `DDL.sql`.
- Los tipos de datos compatibles con PostgreSQL, tales como `INTEGER`, `VARCHAR`, `DATE`, y `BOOLEAN`.
- Las restricciones necesarias, incluyendo valores no nulos, rangos válidos, listas definidas manualmente y formatos de fecha estándar (YYYY-MM-DD).

7.1.2. Configuración por Tabla

Se generaron al menos 150 registros válidos para cada una de las siguientes tablas:

- `persona`
- `pokemon`
- `alimento`
- `limpiadorCuidador`
- `premio`

Cada tabla fue configurada de manera individual usando:

- **Row Number** para generar claves primarias secuenciales.
- **Custom List** para atributos con valores restringidos (por ejemplo, especies de Pokémon, unidades de medida, facultades).
- **Date** con formato ISO para los campos de fecha.
- **Boolean** para atributos como shiny.

7.1.3. Exportación

- Los datos fueron exportados en formato SQL, desactivando la opción `Include CREATE TABLE` con el fin de evitar duplicar definiciones ya presentes en `DDL.sql`.
- Cada archivo generado contiene únicamente instrucciones `INSERT INTO`, listas para ejecutarse directamente sobre PostgreSQL.
- Los archivos exportados se integraron en el script `DML.sql`, encargado de poblar las tablas definidas en `DDL.sql`.
- Dicho script puede ejecutarse desde herramientas como **pgAdmin**, **DBeaver**, o mediante la terminal `psql`.

7.2. Validación de Datos Generados

Se verificó que los datos cumplieran con todas las restricciones del modelo, incluyendo:

- Formato correcto de las fechas.
- Ausencia de valores nulos en campos obligatorios.
- Coherencia entre claves foráneas (por ejemplo, `idPersona` en `pokemon` corresponde a un registro existente en `persona`).

Además, se diseñaron los datos para garantizar resultados válidos en consultas específicas del archivo `Query.sql`, tales como:

- Participantes cuyo nombre inicia con "R".
- Alimentos caducados dentro de un intervalo de fechas.
- Cuidadores nacidos en noviembre.
- Pokemones con especie `Jigglypuff` y CP = 818.
- Pokemones shiny con especies distintas.

Los datos sintéticos fueron elaborados cuidadosamente para que estas consultas retornen resultados reales y significativos.

Capítulo 8

Consultas, Procedimientos y Disparadores

8.1. Consultas

- Porcentaje de participantes por equipo.
- Pokémon más usados en los enfrentamientos.
- Ganadores y premios por torneo.
- Honorarios con más capturas de Pokémon shiny.
- Cantidad de impuestos recabados por cada alimento.
- Tiempo promedio de estancia de los visitantes por evento.
- Horarios con más capturas.
- Pokémon capturados por evento con porcentaje.
- Ranking de facultades por premios ganados.
- Participantes con Pokémon multitipo.
- Nombre completo de los participantes y su facultad tanto en el torneo de distancia recorrida como en el de captura de shinys, cuya distancia total recorrida sea mayor al promedio de distancia de todos los participantes y además que su número de capturas de shinys sea mayor a 5.
- Listar a los vendedores cuyo total de alimentos vendidos (número de productos distintos que ofrecen) sea mayor a 3.
- Mostrar las facultades que tienen más de 5 participantes inscritos en cualquier torneo.
- Listar los Pokémon Shinys que fueron capturados durante el evento, únicamente si fueron capturados entre las 14:00 y las 18:00.

- Obtener la lista de participantes que estén inscritos en el Torneo de Captura de Shiny y que no estén inscritos en el torneo de distancia recorrida.
- Calcular cuántos Pokémon registró cada participante para el torneo de peleas por cada una de las ediciones.
- Calcular la distancia total recorrida por cada participante en el torneo de distancia recorrida.
- Mostrar a todos los vendedores junto con los alimentos que venden, indicando el precio sin IVA y el precio final.
- Mostrar el nombre completo de todos los participantes junto con su cuenta de Pokémon Go.
- Listar todos los Pokémon cuya especie contiene la cadena "chu".

8.2. Procedimientos y Disparadores

8.2.1. Procedimientos

- Consultar edad de un participante con `idPersona = 5`.
- Consultar cuántos Shinys tiene el participante con `idPersona = 5`.
- Consultar el promedio de CP de los Pokémon del participante con `idPersona = 5`.

8.2.2. Disparadores

- Validar los horarios de entrada y salida de los visitantes, asegurando que la hora de salida sea mayor a la de entrada.
- Guardar en una tabla de registro cada vez que ciertos datos se modifican, permitiendo rastrear quién hizo un cambio y cuándo lo hizo.
- Realizar el conteo de Shinys y mantener consistencia entre las tablas `Pokemon` y `Participante`; al insertar un Shiny, se recalcula automáticamente cuántos Shinys tiene el participante.

Capítulo 9

Conclusión

El desarrollo de este proyecto permitió integrar de manera práctica diversos elementos fundamentales del diseño y manejo de bases de datos relacionales.

La generación de datos sintéticos mediante **Mockaroo** facilitó la construcción de un entorno de prueba realista y controlado, mientras que la implementación de consultas complejas, procedimientos almacenados y disparadores reforzó el entendimiento de la lógica interna y las reglas de integridad del sistema.

En conjunto, el proyecto no solo permitió validar la correcta estructura del modelo relacional, sino también comprobar el funcionamiento de los mecanismos que automatizan, aseguran y enriquecen la manipulación de los datos.

Este ejercicio consolidó habilidades esenciales para el análisis, gestión y operación de bases de datos en escenarios reales.