Universidad Nacional Autónoma de México
Fundamentos de Bases de Datos 2026-1
Práctica 05
Equipo Excelovers • Octubre de 2025

Torneo de Pokémon Go

DDL

1. Correcciones

Registrar:

Cambiamos la participación de la relación Registrar entre encargado registro y ParticipanteUNAM total de lado de participanteUNAM pues debe ser registrado por un encargad y parcial de lado de encargado registro pues no forzosamente ha registrado a alguien.

Llave numcuenta:

Modificamos la llave primaria de ParticipanteUNAM a IdPersona.

Inscribir

Renombramos la tabla Inscribir de la relación entre EncargadoRegistro y Participante UNAM a EncargadoInscribirParticipante y la tabla de la relación entre Evento y ParticipanteUNAM a ParticipanteInscribirEvento.

Registrar distancia recorrida:

La relación registrar entre DistanciaRecorrida y CuentaPokemonGo al ser de uno a muchos. Por lo que se agrega a la tabla DistanciaRecorrida la llave de CuentaPokemonGo quedando: DistanciaRecorrida(edición, IdTorneo, IdDistancia, locación, fecha, hora, IdPersona, CodigoDeEntrenador)

Registrar captura:

La relación registrar entre Pokemon, CapturaPokemon y CuentaPokemonGo debe tener las llaves de las tres entidades quedando Registrar(IdPokemon, edición, IdTorneo, IdCaptura, IdPersona, CodigoDeEntrenador)

2. Dominios:

- **Evento**
  - Edición : Será de tipo Integer y deberá cumplir las restricciones de NOT NULL y UNIQUE ya que es la llave primaria.
  - Fecha: Es de tipo Date y deberá cumplir la restricción NOT NULL.
- **CuentaPokemon**
  - id_persona: Es de tipo Integer ya que hace referencia a la columna id_persona de ParticipanteUNAM. Ya que es parte de la llave primaria debe cumplir con la condición de NOT NULL pero no necesariamente es única pues un participante puede tener multiples cuentas.

- CodigoDeEntrenador: Es de tipo Integer, al ser componente de la llave primaria debe cumplir con ser NOT NULL y como es la llave de la entidad débil debe cumplir con UNIQUE. Para asignar estos códigos automáticamente agregamos Serial.
- Equipo: Será de tipo VARCHAR(20), es decir, una cadena máximo 20 caracteres, toda cuenta debe pertenecer a un equipo por lo que debe cumplir la condición de NOT NULL
- Nivel: Es de tipo smallint, además que toda cuenta debe tener un nivel y estos van desde 0 por lo que debe cumplir las restricciones NOT NULL y CHECK (Nivel >= 1).
- NombreDeUsuario: Es de tipo VARCHAR(30), es decir, una cadena de a lo más 30 caracteres y como toda cuenta requiere de un nombre será de tipo NOT NULL, además consideramos que todas las cuentas deben tener un nombre único por lo que agregamos la restricción UNIQUE.

- **ParticipanteUNAM**
  - **IdPersona:** Es de tipo `INTEGER`, ya que funciona como identificador único de cada participante. Al ser la llave primaria debe cumplir con las restricciones `NOT NULL` y `UNIQUE`, asegurando que no existan valores nulos ni duplicados.
  - **NumeroDeCuenta:** Es de tipo `INTEGER`, pues representa un número identificador institucional. Este campo debe cumplir con las restricciones `NOT NULL` y `UNIQUE`, ya que cada participante tiene un número de cuenta único.
  - **Nombre:** Se define como `VARCHAR(20)` porque representa una cadena de texto de hasta 20 caracteres. Todo participante debe tener un nombre registrado, por lo que cumple la restricción `NOT NULL`.
  - **ApellidoMaterno:** Es de tipo `VARCHAR(20)` y cumple con `NOT NULL`, ya que forma parte del nombre completo del participante y se requiere para su identificación.
  - **ApellidoPaterno:** También de tipo `VARCHAR(20)` y con restricción `NOT NULL`, por las mismas razones que el campo anterior.
  - **FechaNacimiento:** De tipo `DATE`, porque almacena fechas. Todo participante debe tener una fecha de nacimiento registrada, por lo que este atributo es `NOT NULL`.
  - **Sexo:** Es de tipo `VARCHAR(10)` y tiene una restricción `CHECK (Sexo IN ('M', 'H', 'Otro'))`, asegurando que los valores posibles correspondan solo a las tres categorías establecidas. Puede admitir nulos si el dato no se especifica.
  - **Carrera:** Se define como `VARCHAR(20)` porque representa el nombre de la carrera del participante. Todo participante debe estar inscrito en una carrera, por lo que cumple la restricción `NOT NULL`.
  - **Facultad:** Es de tipo `VARCHAR(20)` y representa la facultad a la que pertenece el participante. Este dato es obligatorio, por lo tanto cumple `NOT NULL`.

- **Espectador**
  - **IdPersona:** Es de tipo `INTEGER`, ya que identifica de forma única a cada espectador. Al ser la llave primaria, cumple con las restricciones `NOT NULL` y `UNIQUE`, garantizando que no existan valores nulos ni repetidos.
  - **Nombre:** Se define como `VARCHAR(20)`, es decir, una cadena de texto de hasta 20 caracteres. Todo espectador debe tener un nombre registrado, por lo que cumple la restricción `NOT NULL`.
  - **ApellidoMaterno:** Es de tipo `VARCHAR(20)` y representa el apellido materno del espectador. Se establece la restricción `NOT NULL`, pues se considera un dato obligatorio para su identificación completa.
  - **ApellidoPaterno:** También es de tipo `VARCHAR(20)` y debe cumplir con `NOT NULL`, ya que forma parte de los datos personales requeridos del espectador.
  - **FechaNacimiento:** Es de tipo `DATE`, lo que permite almacenar la fecha de nacimiento. Este dato es obligatorio para el registro, por lo que cumple la restricción `NOT NULL`.
  - **Sexo:** Es de tipo `VARCHAR(10)` y tiene la restricción `CHECK (Sexo IN ('M', 'H', 'Otro'))`, lo cual limita los valores posibles a las tres opciones establecidas. Este campo puede admitir valores nulos si el espectador decide no especificarlo.
  - **HoraIngreso:** Se define como `TIMETZ`, ya que almacena una hora junto con la zona horaria. Representa la hora a la que el espectador entra al evento; puede ser nula en caso de no registrarse su ingreso y además no sabemos en qué momento se llenará este dato.
  - **HoraSalida:** También es de tipo `TIMETZ`, utilizada para almacenar la hora en la que el espectador abandona el evento. Este campo puede ser nulo si el registro de salida no se realiza y además no sabemos en qué momento se llenará este dato.

- **ComprarEspectador**
  - **IdPersona:** Es de tipo `INTEGER`, ya que hace referencia a la llave primaria `IdPersona` de la tabla **Espectador**. Al ser además parte de la llave compuesta de esta entidad, cumple con la restricción `NOT NULL`, pero no necesita ser `UNIQUE`, pues un mismo espectador puede realizar varias compras diferentes.
  - **IdAlimento:** Es de tipo `INTEGER` y funciona como llave foránea que hace referencia a la columna `IdAlimento` de la tabla **Alimento**. Al formar parte de la llave primaria compuesta junto con `IdPersona`, debe cumplir con la condición `NOT NULL`.
  - **MetodoDePago:** Se define como `VARCHAR(20)`, ya que representa una cadena de texto de hasta 20 caracteres que describe el método de pago utilizado. Toda compra debe tener un método de pago especificado, por lo que cumple `NOT NULL`, además de la restricción `CHECK (MetodoDePago IN ('Tarjeta', 'Efectivo', 'Transferencia'))`, que limita los valores a las tres opciones válidas.

  - **Cantidad:** Es de tipo `REAL`, ya que puede incluir valores decimales. Representa la cantidad del producto adquirido, y aunque puede variar o ser opcional dependiendo del contexto, se recomienda que este valor sea positivo, por lo que podría añadirse la restricción `CHECK (Cantidad >0)`.

- **ComprarParticipanteUNAM**

  - **IdPersona:** Es de tipo `INTEGER`, ya que hace referencia a la columna `IdPersona` de la tabla **ParticipanteUNAM**. Al formar parte de la llave compuesta de esta entidad, debe cumplir con la restricción `NOT NULL`. No requiere ser `UNIQUE`, pues un mismo participante puede realizar múltiples compras distintas.
  - **IdAlimento:** Es de tipo `INTEGER` y corresponde a una llave foránea que referencia la columna `IdAlimento` de la tabla **Alimento**. Al formar parte de la llave primaria compuesta, cumple con la restricción `NOT NULL`.
  - **MetodoDePago:** Se define como `VARCHAR(20)` ya que almacena cadenas de texto cortas que especifican el medio de pago utilizado. Toda compra debe tener un método de pago indicado, por lo que cumple con `NOT NULL`, además de la restricción `CHECK (MetodoDePago IN ('Tarjeta', 'Efectivo', 'Transferencia'))` que limita los valores válidos.
  - **Cantidad:** Es de tipo `REAL`, ya que puede incluir valores decimales. Representa la cantidad adquirida por el participante; puede ser opcional o configurable según el caso. Si se desea mayor control, podría añadirse la restricción `CHECK (Cantidad >0)` para asegurar que no existan valores negativos o nulos.

- **CorreoParticipante**

  - **IdPersona:** Es de tipo `INTEGER`, ya que hace referencia a la columna `IdPersona` de la tabla **ParticipanteUNAM**. Al formar parte de la llave compuesta de esta entidad, debe cumplir con la restricción `NOT NULL`. No requiere ser `UNIQUE`, ya que un mismo participante puede tener registrados varios correos distintos.
  - **Correo:** Se define como `VARCHAR(50)`, lo que permite almacenar direcciones de correo electrónico de hasta 50 caracteres. Forma parte de la llave primaria compuesta junto con `IdPersona`, por lo que cumple la restricción `NOT NULL`. Cada combinación de persona y correo debe ser única, garantizando así que no se repitan registros.

- **ParticipanteInscribirEvento**

  - **Edicion:** Es de tipo `INTEGER`, ya que hace referencia a la columna `Edicion` de la tabla **Evento**. Al formar parte de la llave primaria compuesta, debe cumplir con la restricción `NOT NULL`. Representa la edición específica del evento en la que se inscribe el participante.
  - **IdPersona:** Es de tipo `INTEGER`, puesto que hace referencia a la columna `IdPersona` de la tabla **ParticipanteUNAM**. Forma parte de la llave compuesta junto con `Edicion` y, por lo tanto, cumple con la restricción `NOT NULL`. No es `UNIQUE`, ya que un mismo participante puede inscribirse en diferentes ediciones de eventos.

  - **Fecha:** Se define como `DATE`, ya que almacena la fecha en que el participante se inscribió al evento. Toda inscripción debe registrar su fecha correspondiente, por lo que cumple con la restricción `NOT NULL`.
  - **Costo:** Es de tipo `REAL`, ya que puede incluir valores decimales. Representa el monto pagado por la inscripción. Debe cumplir con la restricción `NOT NULL`, y se recomienda añadir una condición adicional `CHECK (Costo >= 0)` para asegurar que los valores sean no negativos.

- **Asistir**

  - **Edicion:** Es de tipo `INTEGER`, ya que hace referencia a la columna `Edicion` de la tabla **Evento**. Al formar parte de la llave primaria compuesta, debe cumplir con la restricción `NOT NULL`. Indica la edición del evento a la que asiste el espectador.
  - **IdPersona:** Es de tipo `INTEGER`, pues hace referencia a la columna `IdPersona` de la tabla **Espectador**. Al formar parte de la llave compuesta, también cumple con la restricción `NOT NULL`. No requiere ser `UNIQUE`, ya que un mismo espectador puede asistir a múltiples ediciones de eventos diferentes.

- **TelefonoParticipante**

  - **IdPersona:** Es de tipo `INTEGER`, ya que hace referencia a la columna `IdPersona` de la tabla **ParticipanteUNAM**. Al formar parte de la llave compuesta, cumple con la restricción `NOT NULL`. No requiere ser `UNIQUE`, pues un mismo participante puede tener registrados varios números telefónicos diferentes.
  - **Telefono:** Se define como `CHAR(10)`, ya que representa un número telefónico con una longitud fija de 10 dígitos. Este campo forma parte de la llave primaria junto con `IdPersona`, por lo que cumple la restricción `NOT NULL`. Cada combinación de participante y número debe ser única, garantizando que no se repitan registros.

- **Pokemon**

  - IdPokemon: Será de tipo Integer, como es la llave primaria debe cumplir con ser NOT NULL y UNIQUE.
  - IdPersona y CodigoDeEntrenador son la llave compuesta de CuentaPokemonGo que estamos usando aquí como llave foránea, ambas son de tipo Integer. Como consideramos que todo Pokemon esta asociado a una cuenta deben cumplir la restricción de NOT NULL.
  - Nombre: Es de tipo VARCHAR(50), es decir, cadenas de máximo 50 caracteres. Todo pokemon debe tener nombre por lo que debe cumplir la restricción de NOT NULL.
  - Sexo: Como usaremos la convención 'M' para denotar macho, 'H' para denotar hembra y 'Otro' para otro caso, usaremos el tipo VARCHAR(10), debe tener uno asignado así que debe cumplir las restricciones NOT NULL CHECK (Sexo IN ('M', 'H', 'Otro')),

  - Peso:Es de tipo real pues puede ser décimal, todos deben deben tener registrado su peso y este valor solo puede positivo así que debe cumplir las restricciones NOT NULL y CHECK (Peso >0).
  - PuntosDeCombate: Son datos de tipo Integer, los cuales solo pueden ser positivos y todo pokemon debe tenerlos asignados así que deben cumplir con las restricciones NOT NULL CHECK (PuntosDeCombate >= 0).
  - Shiny: Este valor se refiere a si un pokemon es o no es shiny por lo que es de tipo Boolean y debe cumplir la restricción de NOT NULL.
  - Tipo: El tipo será especificado en una cadena de tipo VARCHAR(20), es decir, con longitud máxima de 20 caracteres y debe cumplir la reestricción de NOT NULL.
  - Especie: Se especificara con una cadena de máximo 20 caracteres por lo que usamos Varchar(20) y debe cumplir la restricción de NOT NULL

- **TorneoCapturaShinys**
  - **Edicion**: Es una llave foránea referente a la columna `Edicion` de la tabla **Evento**, por lo que es de tipo `INTEGER`. Como además es parte de la llave primaria debe cumplir con ser `NOT NULL`.
  - **IdTorneo**: Es de tipo `INTEGER` y es parte de la llave primaria y además la llave de la entidad débil, por lo que además de cumplir la restricción de `NOT NULL` cumple con ser `UNIQUE`.
  - **IdPersona**: Al ser una llave foránea de **ParticipanteUNAM**, respetamos su tipo `INTEGER`. Este dato hace referencia al ganador del torneo por lo que puede ser `NULL`.
  - **CantidadAPremiar**: Esta información al referirse a dinero será de tipo `REAL` y según lo especificado en el caso de uso tendrá el valor de `DEFAULT 500.0`.

- **TorneoDistanciaRecorrida**
  - **Edicion**: Al ser una llave foránea a `Edicion` de **Evento**, respeta el tipo `INTEGER` y como es parte de la llave compuesta debe cumplir la restricción `NOT NULL`.
  - **IdTorneo**: Es de tipo `INTEGER` y al ser la llave primaria de la entidad cumple `NOT NULL` y `UNIQUE`.
  - **IdPersona**: Es una llave foránea a **ParticipanteUNAM(IdPersona)** por lo que es de tipo `INTEGER`. Al reflejar al ganador puede ser `NULL` en algunos casos.
  - **CantidadAPremiar**: Usamos `REAL` ya que el monto del premio podría incluir centavos, además asignamos `DEFAULT 500.0`, que es lo que especifica el caso de uso.

- **CapturaPokemon**
  - **Edicion**: Es de tipo `INTEGER`, como es parte de la llave compuesta debe cumplir con ser `NOT NULL`.

  - **IdTorneo**: Es de tipo `INTEGER`, como es parte de la llave compuesta debe cumplir con ser `NOT NULL`.
  - **IdCaptura**: Es de tipo `INTEGER`, como es parte de la llave compuesta debe cumplir con ser `NOT NULL`, además como es la llave primaria de la entidad débil debe cumplir con la restricción de `UNIQUE`.

- **Registrar**
  - **IdPokemon**: Es una llave foránea de tipo `INTEGER`. Como todo registro de una captura Pokémon necesita especificar un Pokémon, cumple la restricción de `NOT NULL`.
  - **CodigoDeEntrenador**, **IdPersona**: Son la llave compuesta de **CuentaPokemonGo**, por lo que son de tipo `INTEGER` y, como para registrar una captura de Pokémon se requiere una cuenta, cumplen la restricción `NOT NULL`.
  - **IdCaptura**, **Edicion**, **IdTorneo**: Son la llave compuesta de **CapturaPokemon**, por lo que son de tipo `INTEGER` y, como es el objeto que registramos, deben cumplir la restricción `NOT NULL`.
  - **Fecha**: Es de tipo `DATE` y debe cumplir la restricción `NOT NULL`.
  - **Hora**: Es de tipo `TIMETZ` (hora con zona horaria) y debe cumplir la restricción `NOT NULL`.

- **DistanciaRecorrida**
  - **Edicion**, **IdTorneo**: Al ser la llave compuesta de **TorneoDistanciaRecorrida** son de tipo `INTEGER` y, como son parte de la llave compuesta, cumplen la restricción `NOT NULL`.
  - **IdDistancia**: Al ser parte de la llave compuesta y la llave de la entidad débil original, es de tipo `INTEGER` y debe cumplir las restricciones `NOT NULL` y `UNIQUE`.
  - **IdPersona**, **CodigoDeEntrenador**: Al ser la llave foránea de la tabla **ParticipanteUNAM**, usamos tipo de dato `INTEGER` y deben ser `NOT NULL`, pues representan la distancia recorrida por un participante.
  - **Locación**: Este valor es uno de los puntos especificados en el caso de uso, por lo que será de tipo `VARCHAR(10)`, `NOT NULL` y cumplirá con `CHECK (Locación IN ('Universum', 'Entrada', 'Rectoria'))`.
  - **Fecha**: Será de tipo `DATE` y debería ser siempre registrada, por lo que debe ser `NOT NULL`.
  - **Hora**: Es de tipo `TIMETZ` y deberá registrarse siempre, por lo que debe cumplir la restricción `NOT NULL`.

- **ComprarCuidador**
  - **IdPersona**: Se define como `INTEGER` para representar una llave foránea con la tabla **Cuidador**. Se coloca `NOT NULL` porque siempre debe existir un cuidador asociado a la compra.

- **IdAlimento**: Se define como `INTEGER` para representar una llave foránea con la tabla **Alimento**. Se coloca `NOT NULL` para garantizar la existencia del producto comprado.
- **MetodoDePago**: Se define como `VARCHAR(20)`, un texto corto, porque solo se almacenan nombres de métodos de pago, validados por `CHECK (MetodoDePago IN ('Tarjeta', 'Efectivo', 'Transferencia'))`. Se coloca `NOT NULL` porque cada compra debe registrarse con un método de pago.
- **Cantidad**: Se define como `REAL` estrictamente positivo para considerar todos los tipos de precios. Se coloca como `NOT NULL` para asegurar que siempre se registre una cantidad.

- **Limpiador**
  - **IdPersona:** Es de tipo `INTEGER`. Se declara con `NOT NULL` porque no puede quedar sin identificador y con `UNIQUE` para asegurar que no existan duplicados. Es la llave primaria de la tabla.
  - **Nombre:** Se define como `VARCHAR(20)` ya que permite almacenar cadenas de texto de longitud variable hasta 20 caracteres. Se establece con `NOT NULL`.
  - **ApellidoMaterno:** Es de tipo `VARCHAR(20)` y cumple con la restricción `NOT NULL`, ya que forma parte del nombre completo del individuo y es requerido obligatoriamente.
  - **ApellidoPaterno:** También es `VARCHAR(20)` con restricción `NOT NULL`, asegurando que el registro esté completo al incluir ambos apellidos del limpiador.
  - **FechaNacimiento:** Se utiliza el tipo `DATE` ya que este dato representa una fecha específica. Se establece como `NOT NULL`.
  - **Sexo:** Es de tipo `VARCHAR(10)` y tiene la restricción `CHECK (Sexo IN ('M', 'H', 'Otro'))`, limitando los valores posibles a opciones válidas. Se declara `NOT NULL`.
  - **Calle:** Se define como `VARCHAR(20)` con `NOT NULL`, ya que forma parte fundamental del domicilio del empleado.
  - **Colonia:** Es `VARCHAR(20)` y tiene la restricción `NOT NULL`.
  - **Ciudad:** Se utiliza `VARCHAR(20)` y se establece `NOT NULL` para asegurar que el domicilio está correctamente especificado.
  - **CodigoPostal:** Es de tipo `INTEGER` y se declara `NOT NULL`.
  - **NumInterior:** Se define como `INTEGER`. No tiene `NOT NULL` porque no todos los domicilios cuentan con número interior.
  - **NumExterior:** Es `INTEGER` y tiene la restricción `NOT NULL`, ya que representa el número principal del domicilio y es indispensable para su identificación.
  - **Ubicacion:** Se utiliza el tipo `VARCHAR(20)` ya que al momento de registrar al Limbiador puede no tener asignada una ubicación todavía.
  - **Horario:** Se define como `VARCHAR(10)` y contiene un `CHECK (Horario IN ('Matutino', 'Vespertino'))`, restringiendo los valores posibles a turnos permitidos. Puede admitir valores nulos si aún no se asigna un turno.

  - **Salario:** Es de tipo `REAL`, permitiendo valores con decimales. Incluye una restricción `CHECK (Salario >= 0)`.

- **CorreoLimpiador** El atributo correo es multivaluado, ya que un limpiador puede tener varios correos electrónicos. Por esta razón se convierte en una tabla independiente. Además, se establece una llave primaria compuesta por `IdPersona` y `Correo` para evitar duplicados y asegurar que un mismo correo no se repita para la misma persona.

  - **IdPersona**: Se define como `NOT NULL` para mantener la relación con la tabla **Limpiador**.
  - **Correo**: Se define como `NOT NULL` porque no tendría sentido registrar una fila sin el correo correspondiente. Se especifica como una cadena variable `VARCHAR(50)`.

- **TelefonoLimpiador** El atributo teléfono es multivaluado, ya que un limpiador puede tener uno o más números de contacto. Por lo tanto, se separa en una tabla independiente. Además, se establece una llave primaria compuesta por `IdPersona` y `Telefono` con el fin de evitar que el mismo número de teléfono se registre más de una vez para la misma persona.

  - **IdPersona**: Se define como `NOT NULL` para indicar a qué limpiador pertenece el número.
  - **Telefono**: Se registra como `NOT NULL` al ser el valor principal del atributo multivaluado. Se especifica como una cadena fija de 10 caracteres, siguiendo la convención nacional `CHAR(10)`.

- **TrabajarLimpiador** La relación entre **Limpiador** y **Evento** es de muchos a muchos, ya que un limpiador puede participar en varios eventos y un evento puede contar con múltiples limpiadores. Para representar esta relación correctamente en el modelo relacional, se crea una tabla independiente.

  - **IdPersona**: Se define como `NOT NULL` para indicar qué limpiador participa en el evento.
  - **Edicion**: Se establece como `NOT NULL` porque representa el identificador del evento en el cual trabaja el limpiador.

- **ComprarLimpiador**

  - **IdPersona**: Se define como `INTEGER` para representar una llave foránea con la tabla **Limpiador**. Se coloca `NOT NULL` porque siempre debe existir un limpiador asociado a la compra.
  - **IdAlimento**: Se define como `INTEGER` para representar una llave foránea con la tabla **Alimento**. Se coloca `NOT NULL` para garantizar la existencia del producto comprado.
  - **MetodoDePago**: Se define como `VARCHAR(20)`, un texto corto, porque solo se almacenan nombres de métodos de pago, validados por `CHECK (MetodoDePago`

`IN ('Tarjeta', 'Efectivo', 'Transferencia')`). Se coloca `NOT NULL` porque cada compra debe registrarse con un método de pago.

- **Cantidad**: Se define como `REAL` estrictamente positivo, para considerar todos los tipos de precios. Se coloca como `NOT NULL` para asegurar que siempre se registre una cantidad.

- **TorneoPelea**
  - **Edicion:** Es de tipo `INTEGER` y se establece con la restricción `NOT NULL`. Representa el identificador del evento del cual el torneo depende, ya que TorneoPelea es una entidad débil de **Evento**. Este atributo forma parte de la llave primaria compuesta y sirve como llave foránea también.
  - **IdTorneo:** Se declara como `INTEGER` con las propiedades `UNIQUE`. Forma parte de la llave primaria compuesta.
  - **IdPersona:** Es de tipo `INTEGER` y funciona como llave foránea hacia **ParticipanteUNAM** con el objetivo de representar al posible ganador del torneo. No tiene `NOT NULL`, ya que el ganador puede no estar definido al momento de la creación del registro.
  - **CantidadAPremiar:** Es el premio de tipo `REAL` y tiene un valor por defecto de `500.0`.

- **PeleaTorneo**
  - **Edicion:** Se define como `INTEGER` y es `NOT NULL`. Conforma la llave primaria y llave foránea que indica a qué edición de evento pertenece la pelea.
  - **IdTorneo:** Se define como `INTEGER` y es `NOT NULL`. Conforma la llave primaria y llave foránea que identifica el torneo dentro de la edición.
  - **NumeroPelea:** Se define como `INTEGER` y `UNIQUE`. Es `NOT NULL` y funciona como parte principal de la llave primaria compuesta.
  - **IdPersona:** Se define como `INTEGER`. Es llave foránea que identifica al participante que compite en la pelea.
  - **CodigoDeEntrenador:** Se define como `INTEGER` y es `NOT NULL`. Forma parte de la llave foránea hacia la cuenta del participante en Pokémon Go.
  - **Fecha:** Se define como `DATE` y es `NOT NULL`. Indica la fecha en que ocurre la pelea.
  - **Fase:** Se define como `INTEGER` y es `NOT NULL`. Representa la fase del torneo en la que se desarrolla la pelea.

- **Utilizar**
  - **IdPokemon:** Se define como `INTEGER` y es `NOT NULL`. Es llave foránea que indica qué Pokémon fue utilizado en la pelea.
  - **Edicion:** Se define como `INTEGER` y es `NOT NULL`. Es llave foránea que indica la edición del evento en la que se utiliza el Pokémon.
  - **IdTorneo:** Se define como `INTEGER` y es `NOT NULL`. Es llave foránea que identifica el torneo correspondiente a la pelea.

  - **NumeroPelea:** Se define como `INTEGER` y es `NOT NULL`. Es llave foránea que identifica la pelea específica dentro del torneo donde se utiliza el Pokémon.

- **Alimento**
  - **IdAlimento**: Se define como un valor entero (`INTEGER`) que actúa como llave primaria. Debe cumplir con las restricciones `NOT NULL` y `UNIQUE`, garantizando que cada alimento se identifique de manera única.
  - **IdPersona**: Es una llave foránea que referencia a la tabla **Vendedor**, indicando quién ofrece el alimento. Se establece como `NOT NULL` para asegurar la existencia del vendedor asociado.
  - **FechaDeCaducidad**: Indica la fecha de vencimiento del producto. Es de tipo `DATE` y se establece como `NOT NULL`, ya que todos los alimentos deben tener registrada una fecha de caducidad.
  - **Nombre**: Se define como una cadena de texto variable `VARCHAR(20)`, que almacena el nombre comercial del alimento. Se marca como `NOT NULL`.
  - **Tipo**: Se define también como `VARCHAR(20)` para clasificar el alimento. Se requiere que sea `NOT NULL`.
  - **Precio**: Representa el costo del alimento. Es de tipo `REAL`, debe cumplir con las restricciones `NOT NULL` y `CHECK(Precio >0)` para garantizar valores positivos.

- **EncargadoRegistro**
  - **IdPersona**: Se define como un valor entero (`INTEGER`) que actúa como llave primaria. Debe cumplir con las restricciones `NOT NULL` y `UNIQUE`, garantizando que cada encargado se identifique de manera única.
  - **Nombre**: Es una cadena de tipo `VARCHAR(20)` que contiene el nombre del encargado. Debe cumplir la restricción `NOT NULL`.
  - **ApellidoPaterno** y **ApellidoMaterno**: Son de tipo `VARCHAR(20)` y también se establecen como `NOT NULL`. Forman parte de la identificación.
  - **FechaDeNacimiento**: Es de tipo `DATE` y se establece como `NOT NULL`.
  - **Sexo**: Es de tipo `VARCHAR(10)`; debe cumplir con las restricciones `CHECK(Sexo IN ('M', 'H', 'Otro'))` y `NOT NULL`.
  - **Calle**, **Colonia** y **Ciudad**: Son de tipo `VARCHAR(20)` y se establecen como `NOT NULL`, ya que describen la dirección del encargado.
  - **CodigoPostal**: Es de tipo `INTEGER NOT NULL` y representa el código postal de su domicilio.
  - **NumInterior** y **NumExterior**: Son de tipo `INTEGER` y deben cumplir con `NOT NULL` para completar la dirección.
  - **EsJugador**: Es un valor `BOOLEAN NOT NULL` que indica si el encargado también participa como jugador en algún evento.

- **ComprarEncargadoRegistro**

  - **IdPersona**: Llave foránea que referencia a la tabla **EncargadoRegistro**, establecida como `NOT NULL`.
  - **IdAlimento**: Llave foránea que referencia a la tabla **Alimento**, también `NOT NULL`.
  - **MetodoDePago**: Se define como `VARCHAR(20)` con las restricciones `CHECK (MetodoDePago IN ('Tarjeta', 'Efectivo', 'Transferencia'))` y `NOT NULL`.
  - **Cantidad**: Es de tipo `REAL NOT NULL CHECK(Cantidad >0)`, representando la cantidad de unidades adquiridas.

- **CorreoEncargadoRegistro** El atributo correo es multivaluado, ya que un encargado puede tener varios correos electrónicos. Por esta razón se convierte en una tabla independiente.

  - **IdPersona**: Llave foránea que referencia a la tabla **EncargadoRegistro**, establecida como `NOT NULL`.
  - **Correo**: Es el valor principal del atributo multivaluado. Se define como `VARCHAR(50) NOT NULL`.

  La llave primaria es compuesta por `(IdPersona, Correo)` para evitar que el mismo correo se registre más de una vez para la misma persona.

- **TelefonoEncargadoRegistro** El atributo teléfono es multivaluado, ya que un encargado puede tener uno o varios números de contacto. Se crea una tabla separada para manejar estos valores.

  - **IdPersona**: Llave foránea que referencia a **EncargadoRegistro**, establecida como `NOT NULL`.
  - **Telefono**: Se define como `CHAR(10) NOT NULL`, siguiendo la convención nacional de diez dígitos.

  La llave primaria está compuesta por `(IdPersona, Telefono)` para evitar duplicados.

- **EncargadoInscribirParticipante**

  - **IdPersona_encargado**: Llave foránea que referencia a **EncargadoRegistro**, marcada como `NOT NULL`.
  - **IdPersona_participante**: Llave foránea que referencia a **ParticipanteUNAM**, también `NOT NULL`.
  - **Fecha**: Es de tipo `DATE NOT NULL` e indica el día en que se efectuó la inscripción.

- **TrabajarEncargadoRegistro**

  - **Edicion**: Llave foránea que referencia a la tabla **Evento**, establecida como `NOT NULL`.
  - **IdPersona**: Llave foránea que referencia a la tabla **EncargadoRegistro**, también `NOT NULL`.

- **TrabajarCuidador**
  - **Edicion**: Llave foránea que referencia a la tabla **Evento**, establecida como `NOT NULL`.
  - **IdPersona**: Llave foránea que referencia a la tabla **Cuidador**, también `NOT NULL`.
- **Vendedor** Tabla para registrar a las personas encargadas de ventas.
  - **IdPersona**. Se define como `INTEGER` y `NOT NULL`. Es la llave primaria de la tabla, identifica de forma única a cada vendedor.
  - Nombre. Se define como `VARCHAR(20)` para almacenar nombres cortos. Debe cumplir la restricción `NOT NULL` si no se captura.
  - **ApellidoMaterno** y **ApellidoPaterno**. Se definen como `VARCHAR(20)`. Deben de ser `NOT NULL`.
  - **FechaNacimiento**. Se define como `DATE` para registrar la fecha de nacimiento.
  - **Sexo**. Se define como `VARCHAR(10)` para valores breves (*Hombre, Mujer, Otro*). Opcional; se puede validar con `CHECK`.
  - **Calle**, **Colonia**, **Ciudad**. Se definen como `VARCHAR(20)` para datos de domicilio.
  - **CodigoPostal**. Se define como `INTEGER`.
  - **NumInterior**, **NumExterior**. Se definen como `INTEGER`. El exterior es `NOT NULL`.
  - **Ubicacion**. Se define como `VARCHAR(20)` para la sede.
- **ComprarVendedor** Tabla de intersección para compras de *Alimento* realizadas por vendedores; modela una relación muchos-a-muchos.
  - **IdPersona**. `INTEGER NOT NULL`. Llave foránea a **Vendedor**; indica quién realiza la compra.
  - **IdAlimento**. `INTEGER NOT NULL`. Llave foránea a **Alimento**; asegura que el producto exista.
  - **MetodoDePago**. `VARCHAR(20)`. Texto corto para el método de pago. (Restricción `NOT NULL` + `CHECK` con valores válidos como *Tarjeta/Efectivo/Transferencia*.)
  - **Cantidad**. `REAL`. Monto/cantidad adquirida.
    (Recomendable `NOT NULL` + `CHECK(Cantidad >0)`.)
- **CorreoVendedor** El correo es un atributo multivaluado: un vendedor puede tener varios correos. Se separa en una tabla propia con llave primaria compuesta para evitar duplicados por persona.
  - **IdPersona**. `INTEGER NOT NULL`. Llave foránea a *Vendedor*; indica a quién pertenece el correo.
  - **Correo**. `VARCHAR(50) NOT NULL`. Parte de la llave primaria compuesta (*IdPersona, Correo*) para impedir que un mismo correo se repita para la misma persona.
- **TelefonoVendedor** El teléfono también es multivaluado. Se normaliza en una tabla independiente con llave primaria compuesta para evitar duplicidad por persona.

  - **IdPersona**. `INTEGER NOT NULL`. Llave foránea a **Vendedor**.
  - **Telefono**. `CHAR(10) NOT NULL`. Se usa longitud fija de `10` dígitos siguiendo la convención nacional. Forma la llave primaria compuesta.

- **TrabajarVendedor** Representa la relación muchos-a-muchos entre *Vendedor* y *Evento*: un vendedor puede trabajar en varias ediciones y una edición puede tener múltiples vendedores.
  - **Edicion**. `INTEGER NOT NULL`. Identifica la edición del evento; llave foránea a *Evento*.
  - **IdPersona**. `INTEGER NOT NULL`. Identifica al vendedor asignado; llave foránea a *Vendedor*.

- **Cuidador**
  - **IdPersona**: `INTEGER NOT NULL`. Llave primaria de la tabla; identifica de forma única a cada cuidador.
  - **Nombre**: `VARCHAR(20) NOT NULL`. Nombre(s) del cuidador.
  - **ApellidoMaterno**, **ApellidoPaterno**: `VARCHAR(20) NOT NULL`. Forman parte de su identificación.
  - **FechaNacimiento**: `DATE`. Fecha de nacimiento.
  - **Sexo**: `VARCHAR(10)` para valores breves; se puede validar con `CHECK (Sexo IN ('M', 'H', 'Otro'))`.
  - **Calle**, **Colonia**, **Ciudad**: `VARCHAR(20)`. Campos de domicilio.
  - **NumInterior**, **NumExterior**: `INTEGER`. Con restricción `NOT NULL` en **NumExterior**.
  - **Horario**: `VARCHAR(10)`. Texto corto con el turno o horario.
  - **Salario**: `REAL`. Monto del salario; recomendable validar no negatividad con `CHECK (Salario >= 0)`.

- **CorreoCuidador**
  - **IdPersona**: `INTEGER NOT NULL`. Llave foránea a **Cuidador**.
  - **Correo**: `VARCHAR(50) NOT NULL`. Parte de la llave primaria compuesta (`IdPersona, Correo`) para impedir correos repetidos por persona.

- **TelefonoCuidador**
  - **IdPersona**: `INTEGER NOT NULL`. Llave foránea a **Cuidador**.
  - **Telefono**: `CHAR(10) NOT NULL`. Longitud fija de `10` dígitos. Con **IdPersona** integra la llave primaria compuesta.

3. Backup

Restauramos el backup del archivo `DDL.sql` de la forma:

```
$ sudo docker exec -it 23cb57e09241 createdb -U postgres ejemploBackUp
```

```
$ sudo docker exec -i 23cb57e09241 psql -U postgres -d
ejemploBackUp < /home/julietanilem/Descargas/DDL.sql
```

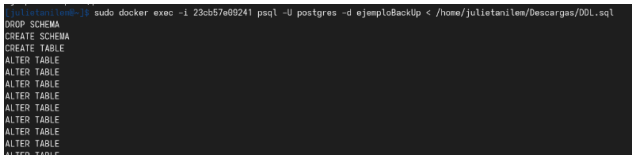


Figura 1: Comando usado para hacer el primer backup

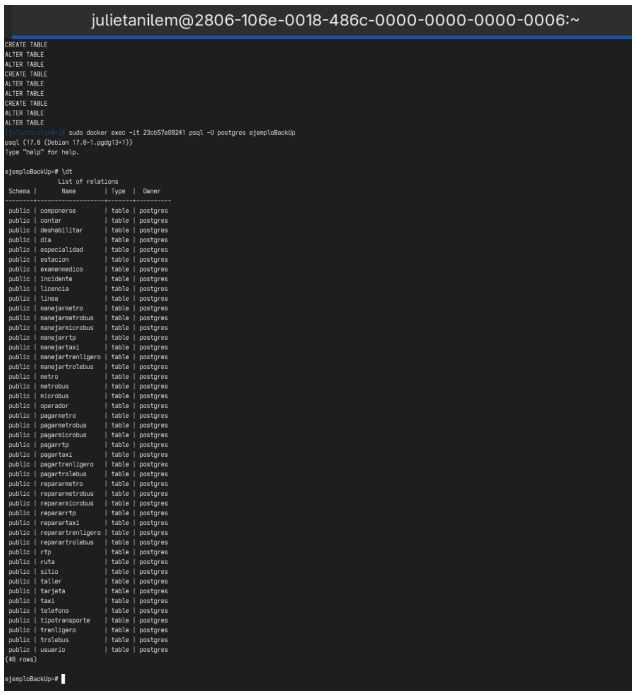


Figura 2: Evidencia de la restauración del backup de ejemplo DDL.sql

Restauramos el backup del archivo `ejemplo.backup` de la forma:

```
$ sudo docker exec -it 23cb57e09241 createdb -U postgres ejemploBackUp2
```

```
$ sudo docker exec -i 23cb57e09241 pg_restore --verbose --clean --no-acl --no-owner
-U postgres -d ejemploBackUp2 < /home/julietanilem/Descargas/ejemplo.backup
```

Figura 3: Comando para hacer el segundo backup

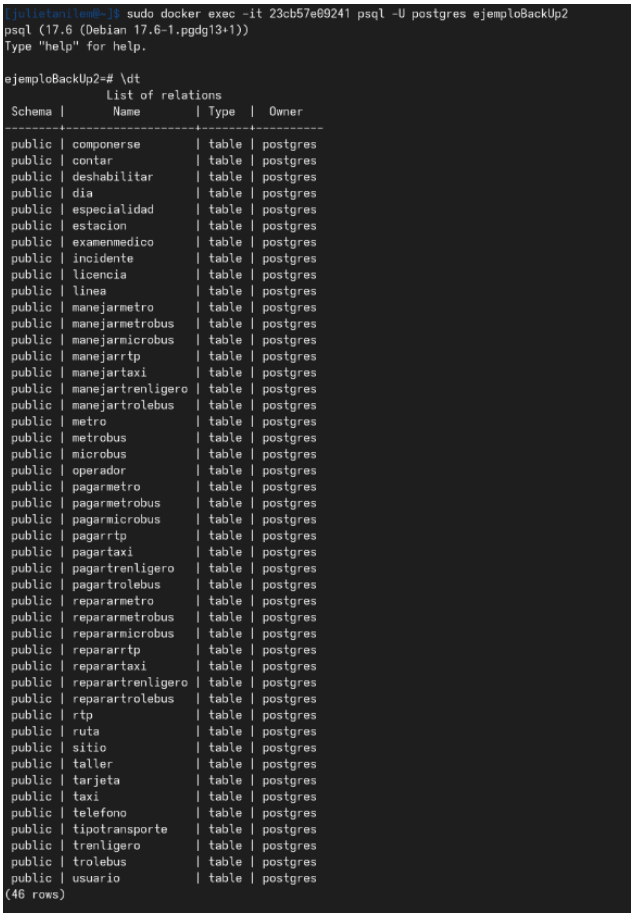


Figura 4: Evidencia del segundo backup

Finalmente usamos los comandos para generar nuestra base:

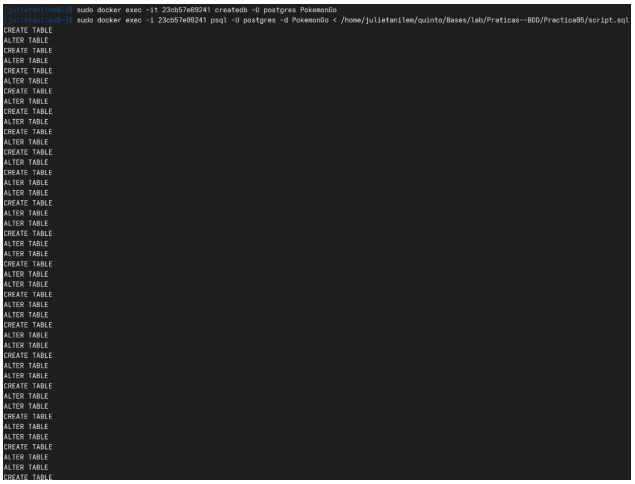
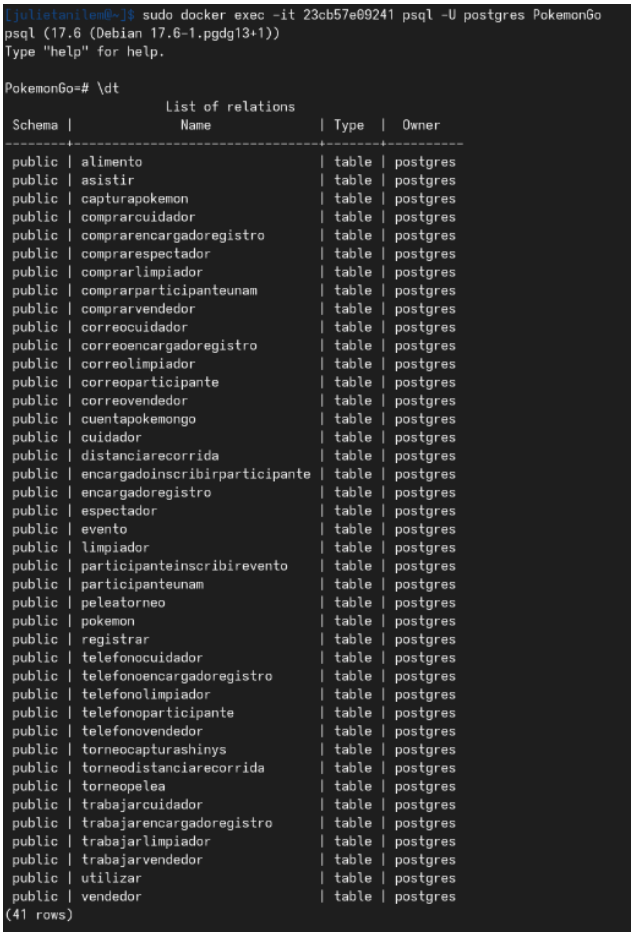


Figura 5: Backup de nuestra base



```
[julietanilem@~]$ sudo docker exec -it 23cb57e09241 psql -U postgres PokemonGo
psql (17.6 (Debian 17.6-1.pgdg13+1))
Type "help" for help.

PokemonGo=# \dt
                  List of relations
 Schema |             Name              | Type  |  Owner
--------+-------------------------------+-------+----------
 public | alimento                      | table | postgres
 public | asistir                       | table | postgres
 public | capturapokemon                | table | postgres
 public | comprarcuidador               | table | postgres
 public | comprarencargadoregistro      | table | postgres
 public | comprarespectador             | table | postgres
 public | comprarlimpiador              | table | postgres
 public | comprarparticipanteunam       | table | postgres
 public | comprarvendedor               | table | postgres
 public | correocuidador                | table | postgres
 public | correoencargadoregistro       | table | postgres
 public | correolimpiador               | table | postgres
 public | correoparticipante            | table | postgres
 public | correovendedor                | table | postgres
 public | cuentapokemongo               | table | postgres
 public | cuidador                      | table | postgres
 public | distanciarecorrida            | table | postgres
 public | encargadoinscribirparticipante | table | postgres
 public | encargadoregistro             | table | postgres
 public | espectador                    | table | postgres
 public | evento                        | table | postgres
 public | limpiador                     | table | postgres
 public | participanteinscribirevento   | table | postgres
 public | participanteunam              | table | postgres
 public | peleatorneo                   | table | postgres
 public | pokemon                       | table | postgres
 public | registrar                     | table | postgres
 public | telefonocuidador              | table | postgres
 public | telefonoencargadoregistro     | table | postgres
 public | telefonolimpiador             | table | postgres
 public | telefonoparticipante          | table | postgres
 public | telefonovendedor              | table | postgres
 public | torneocapturashinys           | table | postgres
 public | torneodistanciarecorrida      | table | postgres
 public | torneopelea                   | table | postgres
 public | trabajarcuidador              | table | postgres
 public | trabajarencargadoregistro     | table | postgres
 public | trabajarlimpiador             | table | postgres
 public | trabajarvendedor              | table | postgres
 public | utilizar                      | table | postgres
 public | vendedor                      | table | postgres
(41 rows)
```

Figura 6: Resultado final de nuestra base