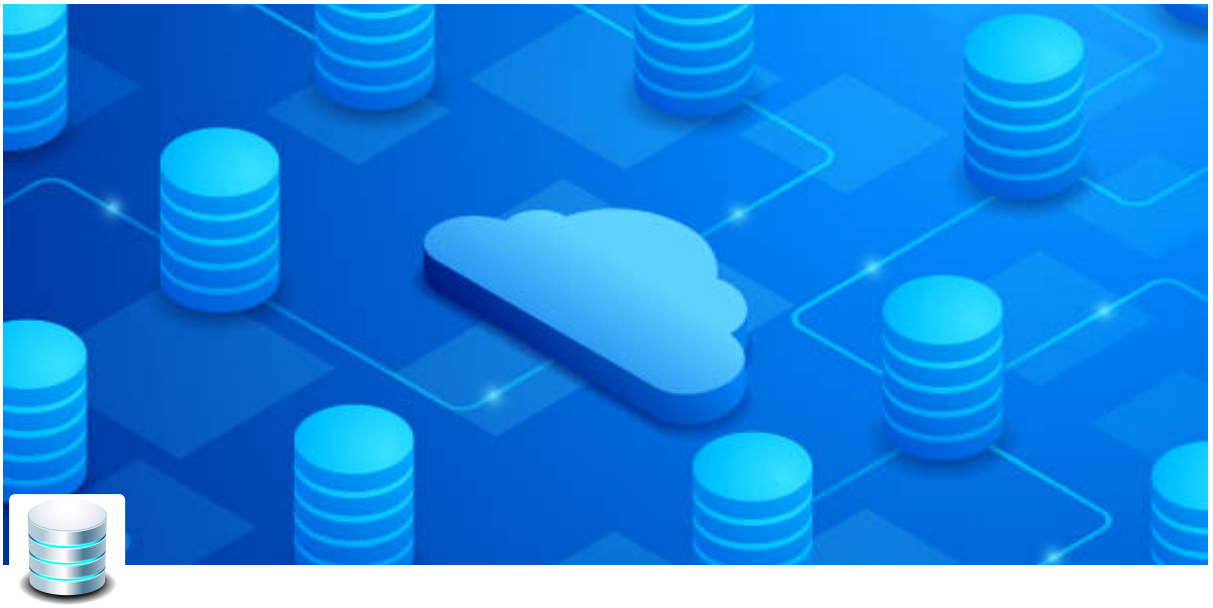

Notes Data Base

Problemas y Variantes en el Diseño de la Base de Datos

Problema No1: Gestión de Roles

Existen 2 problemas que estamos tratando de resolver de manera eficiente:

No1: Cómo se llevaría a cabo la gestión de roles.
No2: Si un deck puede pertenecer a más de una persona.

Con respecto al No1, tenemos 3 variantes:

Variante 1:

- **Entidades:**
  - Usuario (IdUsuario,.... Atributos)
  - Torneo (IdTorneo,.... Atributos)
- **Relaciones:**
  - Crear (IdUsuario, IdTorneo, Otros)
  - Competir (IdUsuario, IdTorneo, Otros)

Variante 2:

- **Entidades:**
  - Usuario (IdUsuario,.... Atributos)
  - Rol (IdRol,....Atributos)
  - Torneo (IdTorneo,.... Atributos)
- **Relaciones:**
  - Crear (IdUsuario, IdTorneo, Otros)
  - Competir (IdUsuario, IdTorneo, Otros)
  - Pertenecer (IdUsuario, IdRol)

Variante 3:

- **Entidades:**
  - Usuario (IdUsuario,.... Atributos)
  - Administrador (IdUsuario,...Atributos)
  - Jugador (IdUsuario,.... Atributos)
  - Torneo (IdTorneo,.... Atributos)
- **Relaciones:**
  - Crear (IdAdmin, IdTorneo, Otros)
  - Competir (IdPlayer, IdTorneo, Otros)

Pros y Contras de las Variantes para la Gestión de Roles:

Variante 1: (Eficiente si no se desean tener más de 2 roles)

- **Pros:**
  - Cualquier usuario puede tener el rol de administrador de torneos (gestionar torneos y tener acceso a información única sobre el torneo que crea).
  - Simplifica la estructura de la base de datos al tener una sola entidad de usuario.
  - Bajo costo de almacenamiento.
- **Contras:**
  - No tenemos un sistema cerrado de roles, por lo que si deseamos agregar otros roles, habría que realizar posibles ajustes (un usuario puede

desempeñar ambos roles).

- **Posibles Ajustes:**

  Si fuera una exigencia del cliente que el rol de administrador pudiera ser desempeñado por un grupo cerrado de personas, entonces podríamos hacer lo siguiente:

  Puedes agregar un atributo adicional en la entidad "Usuario" para indicar su rol. Por ejemplo, podrías agregar un campo llamado "EsAdmin" (o similar) que sea una bandera booleana (verdadero o falso) para representar si un usuario es un administrador o no.

  A continuación, te muestro cómo se vería esta modificación en la Variante 1:

  Entidades:

  1. **Usuario (User):**
     - IdUsuario (ID)
     - Otros atributos (nombre, correo electrónico, etc.)
     - EsAdmin (booleano, que indica si el usuario es administrador o no)
  2. **Torneo (Tourney):**
     - IdTorneo (ID)
     - Otros atributos del torneo

  Relaciones:

  - Crear (IdUsuario, IdTorneo, Otros): Un usuario (ya sea administrador o no) puede crear un torneo.
  - Competir (IdUsuario, IdTorneo, Otros): Un usuario (ya sea administrador o no) puede competir en un torneo.

  Si el proyecto contempla la existencia de más de 2 roles, entonces deberíamos considerar la necesidad de aplicar la Variante 2.

Variante 2: (Eficiente si se desean tener más de 2 roles)

- **Pros:**
  - Esta variante puede ser adecuada a largo plazo si se anticipa la existencia de nuevos roles en el sistema de gestión de torneos.
- **Contras:**

- Mayor costo de almacenamiento.
- Requiere consultas de código para controlar los privilegios, lo que la hace más compleja en términos de complejidad temporal.

Variante 3: (Eficiente si se desea gestionar atributos y funcionalidades específicas para cada tipo de usuario)

- **Pros:**
  - Utiliza una entidad "Usuario" con subentidades "Administrador" y "Jugador" para representar los roles y ofrece la posibilidad de gestionar atributos específicos de cada rol.
- **Contras:**
  - Ofrece una forma de gestionar diferentes roles, pero puede ser menos eficiente en términos de almacenamiento.

Con respecto al problema No2: Si un deck puede pertenecer a más de una persona

Variante 1:

- **Pros:**
  - Los decks son muy cambiantes, y esta implementación se desvincula del problema de controlar las operaciones que un usuario puede realizar sobre un deck (añadir, eliminar, cambiar nombre, modificar, entre otras). Estas operaciones con esta variante son bastante sencillas, ya que cada deck se gestiona como una entidad distinta.
  - Baja complejidad en cuanto a consultas.
- **Contras:**
  - Mayor costo de almacenamiento.
  - Limita la compartición de decks entre usuarios, aunque esto podría solucionarse.

Posibles Ajustes para la Variante 1:

- Si es importante permitir la compartición de decks entre usuarios, podrías considerar implementarlo haciendo una copia exacta de los atributos con una llave distinta.

- Estamos analizando posibles mejoras para reducir el costo espacial, quizás una combinación entre la variante 1 y 2.

Variante 2:

- **Pros:**
  - Un poco más eficiente en términos de espacio.
- **Contras:**
  - La eficiencia, especialmente, depende de factores humanos, es decir, que el usuario para obtener un deck ya existente utilice alguna vía distinta a crear un nuevo deck o implementar un sistema de control que verifique si un deck ya existe antes de crear uno nuevo. Si un usuario intenta crear un deck que ya existe, el sistema podría asociarlo al deck existente en lugar de crear uno nuevo, para controlar que no haya un alto nivel de datos duplicados
  - Las operaciones sobre los decks pueden ser problemáticas, y habría que buscar formas de gestionarlas eficientemente.
- **Posibles Reajustes para la Variante 2:**
  - Estamos analizando posibles mejoras