Torneos de Yu-Gi-Oh

Chavely González Acosta C312
José Carlos Pendas Rodriguéz C311
Lázaro David Alba Ajete C311
Max Bengochea Moré C311

October 3, 2023

1 Análisis y reformulación enriquecedora de los requerimientos funcionales e informacionales del sistema

1.1 Requerimientos funcionales:

1. **Registro de usuarios:** Los usuarios se registran en la aplicación proporcionando su información personal: nombre completo, municipio, provincia, teléfono (opcional) y dirección.
2. **Registro de Decks:** Los jugadores registraran sus decks (mazos de cartas) asociándolos con su perfil, incluyendo detalles como: nombre del deck, cantidad de cartas en el mazo principal, en el mazo alternativo y en el mazo extra, así como su arquetipo.
3. **Creación de torneos:** Los administradores pueden crear torneos especificando: nombre del torneo, fecha de inicio y dirección, un torneo puede ser creado por un solo administrador.
4. **Solicitud de inscripción en torneos:** Los jugadores solicitan inscribirse en los torneos disponibles, asegurándose de que no puedan inscribirse después de la fecha de inicio del torneo, y con esactamente un deck.
5. **Inscripción en torneos:** Los administradores reciben las solicitudes de inscripción de los jugadores a los torneos y se encargan de inscribirlos en los mismos.
6. **Gestión de Rondas y Partidas:** La aplicación administra las rondas de los torneos, asignando aleatoriamente los emparejamientos.
7. **Registro de los resultados de las partidas:** Los administradores registran los resultados de las partidas, asegurando que una partida de Yu-Gi-Oh! es de 3 a ganar 2.
8. **Consultas y Estadísticas**: La aplicación ofrece una interfaz donde los usuarios pueden realizar consultas y obtener estadísticas, así como exportar los resultados a otros formatos. Las consultas a modelarse son las siguientes:
    - Los n jugadores con más decks en su poder(de mayor a menor).
    - Los n jugadores más populares entre los jugadores(de mayor a menor).
    - La provincia/municipio donde es más popular un arquetipo.
    - El campeón de un torneo.
    - Los n jugadores con más victorias (ordenados de mayor a menor).
    - El arquetipo más utilizado en un torneo dado.
    - La cantidad de veces que los arquetipos han sido el arquetipo del campeón en un grupo de torneos(en un intervalo de tiempo).
    - La provincia/municipio con más campeones(en un intervalo de tiempo)
    - Dado un torneo y una ronda, cuáles son los arquetipos más representados(cantidad de jugadores usándolos)
    - Los n arquetipos más utilizados por al menos un jugador en el torneo(de mayor a menor)
9. **Seguridad**: La aplicación implementa autenticación y autorización para asegurarse de que solo los administradores pueden realizar acciones como crear torneos y registrar resultados de partidas. Además, se deben validar los datos de entrada para evitar errores o datos incorrectos en la base de datos.

1.2 Requerimientos informacionales del sistema:

Con el objetivo de validar las peticiones de los usuarios, así como dar respuesta a las consultas de los mismos, es necesario mantener diversas informaciones en una base de datos:

1. **Usuario:** De los usuarios se deben almacenar: un identificador para cada usuario, el nombre completo, el teléfono, la dirección, el municipio y la provincia.
2. **Deck:** De los decks se deben almacenar: un identificador para cada deck distinto, el nombre del deck, la cantidad de cartas en el mazo principal, en el mazo alternativo y en el mazo extra, el arquetipo y el usuario que lo creó.
3. **Torneo:** De los torneos se debe almacenar: un identificador para cada torneo, nombre del torneo, fecha de inicio, dirección y el administrador que lo creó.

4. **Rol:** Para cada uno de los roles que puede desempeñar un usuario en la aplicación se guarda un identificador y el nombre del rol.

5. **Partida:** De las partidas se debe almacenar: el identificador de cada usuario que participa(2 en total), el identificador del torneo al cual pertenece, la fecha exacta en la cual se realiza, el resultado de la partida(solo hay 4 resultados posibles: "2:1", "2:0", "1:2", "0:2") y la ronda del torneo en la cual se realiza.

6. **Solicitud de inscripción en torneos:** Los jugadores solicitan inscribirse en algún torneo disponible y estas solicitudes se almacenan como pendientes hasta que algún administrador las acepte e iscriba al jugador o las rechase. Para esto es necesario almacenar: el identificador del usuario, el identificador del torneo, el identificador del deck con el cual se desea inscribir, la fecha en la cual ralizó la solicitud y el estado de la solicitud. Los posibles estados son: pendiente, aceptado, rechazado.

7. **Registro de usuario:** Para todos los usuarios se debe almacenar el identificador del usuario relacionado con el identificador de cada rol que este desempeña.

2 Modelo conceptual de la base de datos:

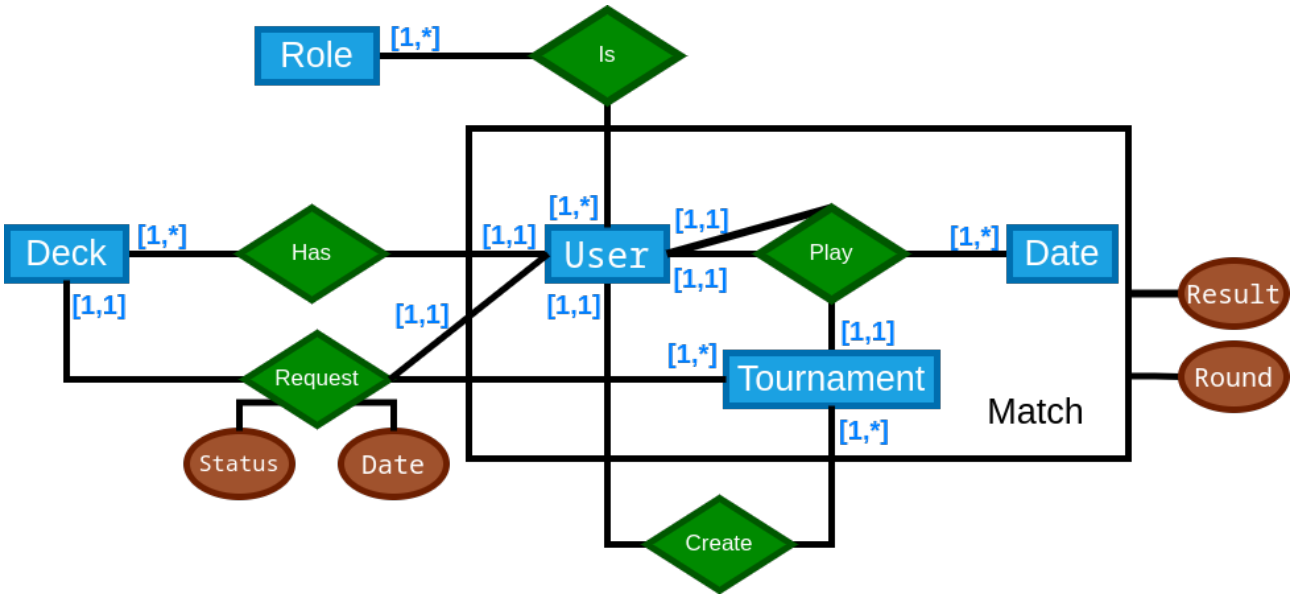

Figure 1: Esquema Relacional

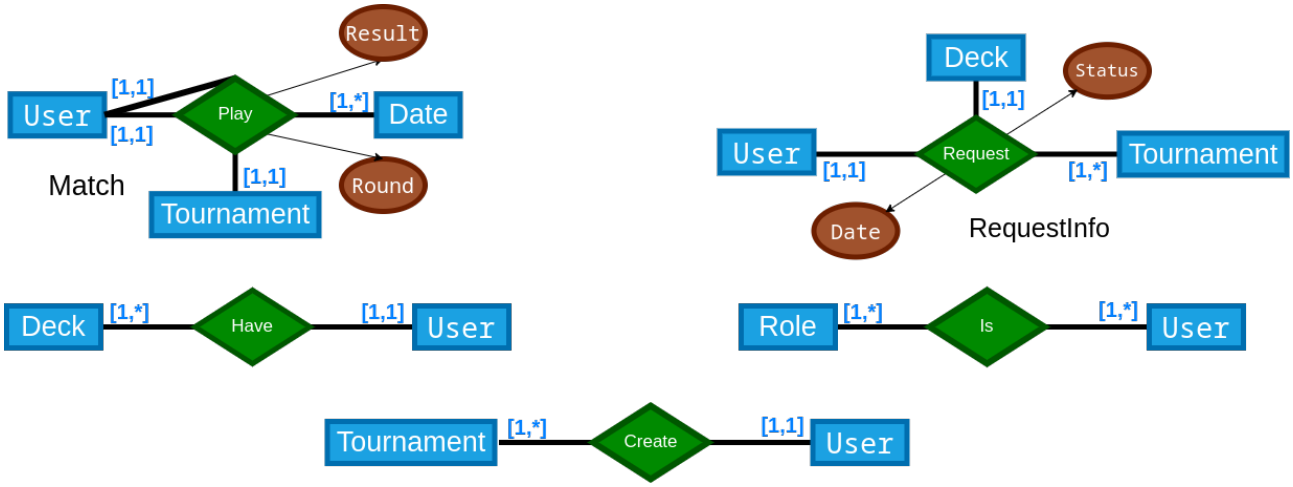

Figure 2: Esquema Relacional separado por relaciones

3 Valoración del diseño de la base de datos en cuanto a si es o no correcto:

Sea el esquema relacional $R(U, F)$:

Conjunto de Atributos(U)

- UserID, user_name, township, province, phone (optional), address, DeckID, deck_name, main_deck_sz, side_deck_sz, extra_deck_sz, archetype, TournamentID, tournament_name, start_date, location, RoleID, role_name, date, result, round, status

Dependencias Funcionales(F)

UserID → [user_name, township, province, phone (optional), address]

DeckID → [RoleID]

DeckID → [deck_name, main_deck_sz, side_deck_sz, extra_deck_sz, archetype]

DeckID → [UserID]

TournamentID → [tournament_name, start_date, location]

RoleID → [role_name]

UserID, TournamentID → [status, date]

UserID, TournamentID → [DeckID]

UserID1, UserID2, TournamentID, date → [IDUser1, IDUser2, IDTournament, Date]

UserID1, UserID2, TournamentID, date → [Result, IDWinner, Round]

Sea $p = \{R1, R2, R3, R4, R5, R6\}$
$R1(U1, F1)$
U1: F1:

DeckID → [deck_name, main_deck_sz, side_deck_sz, extra_deck_sz, archetype]

UserID → [RoleID]

$R2(U2, F2)$ **U2: F2:**

DeckID → [deck_name, main_deck_sz, side_deck_sz, extra_deck_sz, archetype]

DeckID → [UserID]

$R3(U3, F3)$ **U3: F3:**

TournamentID → [tournament_name, start_date, location]

$R4(U4, F4)$ **U4: F4:**

RoleID → [role_name]

$R5(U5, F5)$ **U5: F5:**

UserID, TournamentID → [status, date]

UserID, TournamentID → [DeckID]

$R6(U6, F6)$ **U6: F6:**

UserID1, UserID2, TournamentID, Date → [UserID1, UserID2, TournamentID, date]

UserID1, UserID2, TournamentID, Date → [result, round]

Dados un esquema relacional $R(U, F)$ y una descomposición $\rho = (R1, R2, \ldots, Rk)$, desde el punto de vista teórico, una base de datos relacional está correctamente diseñada o, simplemente, es un diseño correcto si se cumplen las tres propiedades siguientes:

1. Todos los esquemas relacionales $Rj(Uj, Fj)$ (j=1, ..., k) de la descomposición ρ están en 3FN o una forma normal superior.

2. La descomposición ρ satisface la propiedad de join sin pérdida de información (PLJ).

3. La descomposición ρ satisface la propiedad de preservación de dependencias funcionales (PPDF).

Mediante la aplicación del algoritmo pudimos hallar una descomposición ρ que cumple con dos de las características de un diseño correcto. Ahora bien, la tercera condición (la satisfacción de PLJ) se puede comprobar fácilmente aplicando una de las consecuencias del lema que se presenta a continuación,

Consecuencia Si la llave X de $R(U, F)$ está contenida completamente en alguno de los esquemas relacionales de la descomposición, entonces, se puede afirmar directamente que ρ es un diseño correcto En nuestro diserno se cumple con las tres propiedades requeridas las llaves candidatas (IDDeck, IDTournament) o (IDUser, IDTournament) y estas están completamente contenidas en $R5$.

Play	
PK,FK1	**TournamentID**
PK	**Date**
PK,FK2	**UserRID**
PK,FK3	**UserLID**
int	round
Enum	result

User	
PK	**UserID**
str	player_name
str	township
str	province
str	player_address
int	phone_number

Deck	
PK	**DeckID**
str	deck_name
str	archetype
int	main_deck_size (40-60)
int	side_deck_size (0-15)
int	extra_deck_size (0-15)
FK	**UserID**

Tournament	
PK	**TournamentID**
str	tournament_name
str	tournament_address
date	start_date
FK	**UserID**

Is	
PK,FK1	**UserID**
PK,FK2	**RoleID**

Role	
PK	**RoleID**
str	role_name

Request	
PK,FK1	**UserID**
PK,FK2	**TournamentID**
FK3	**DeckID**
date	request_date
Enum	status