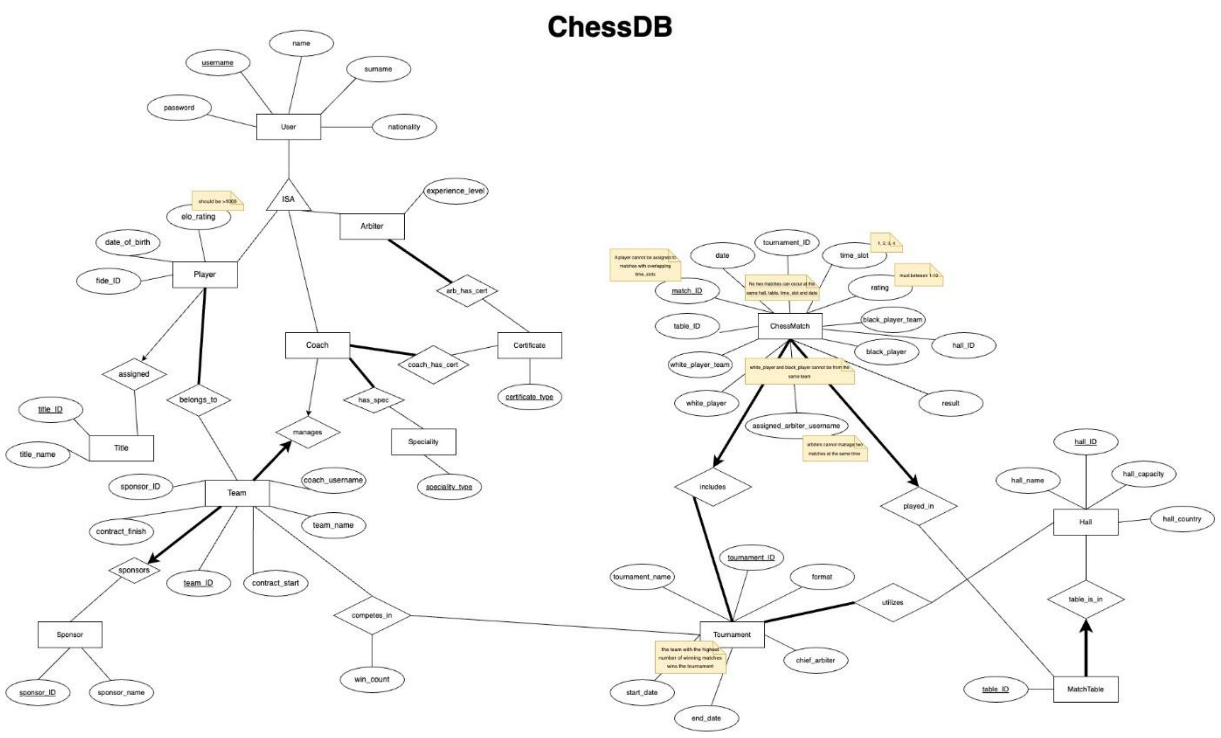
Umut Şendağ 2021400072
Ali Gökçek 2021400012

CMPE 321 Project 1 Part 1: Conceptual Database Design Report

1. Introduction

This project involves designing **ChessDB**, a database system to manage chess tournaments under FIDE rules. The database must store and link information on players, coaches, arbiters, teams, sponsors, matches, halls, and tournaments, while also capturing business constraints such as time-slot scheduling and match outcomes. Below is our **Entity-Relationship (ER) diagram** representing the conceptual design, followed by a discussion of constraints we could not fully represent at the ER level.



2. Overview of the ER Diagram

Our ER diagram begins with a **User** entity (keyed by *username*), specialized into **Player**, **Coach**, and **Arbiter**. Each subclass has specific attributes (e.g., *elo_rating* for Player; *experience_level* for Arbiters; no additional simple attributes for Coach) and relationships (e.g., a Player belongs to one or more Teams; a Coach manages exactly one Team at a time).

Umut Şendağ 2021400072
Ali Gökçek 2021400012

Teams have sponsors, manage players, and compete in **Tournaments**, which include one or more **Matches**. Each Match is assigned an **Arbiter**, occurs in a specific **Hall** at a particular **table_ID** and **time_slot**, and features a *white_player* and *black_player* from different teams.

Key attributes like *rating* (for match quality) and *time_slot* (1 to 4) appear on the **Match** entity, while constraints (e.g., *elo_rating* > 1000, no two matches at the same table/time/hall) are partially noted but typically require additional enforcement at the logical or application level.

---

3. Main Represented ER Constraints

1. **User–Subclasses**: Disjoint, total specialization (every user is exactly one of Player, Coach, or Arbiter).
2. **Player-Title:** A player can have at most one title.
3. **Team–Coach**: Each team has exactly one coach; a coach cannot manage multiple teams simultaneously.
4. **Team–Sponsor**: One sponsor can sponsor many teams; each team has exactly one sponsor.
5. **Player–Team**: A player can join multiple teams (many-to-many).
6. **Tournament–Match**: Each match belongs to exactly one tournament; a tournament can have many matches.
7. **Match**: Attributes such as *time_slot* (1–4), *rating* (1–10), and results.
8. **Hall–MatchTable**: A hall can have multiple tables; each table belongs to one hall.

---

4. Constraints Not Fully Representable in ER

Despite annotations, some constraints require triggers, CHECK conditions, or procedural logic:

1. **ELO Rating > 1000**: Requires a CHECK constraint in SQL.
2. **Rating Between 1 and 10**: Also enforced with a CHECK constraint.

3. **Time Slots 1–4**: Must be restricted with an enumerated type or CHECK constraint.
4. **No Two Matches in Same Hall/Table/Time/Date**: Requires a composite uniqueness or scheduling logic.
5. **Player Cannot Play Overlapping Matches**: Needs triggers or an application-level check.
6. **Arbiter Cannot Oversee Overlapping Matches**: Also needs triggers or scheduling checks.
7. **White/Black Players from Different Teams**: Must be enforced at the schema/application level.
8. **Team With Most Wins Is Tournament Winner**: Requires an aggregate computation, not purely representable in ER.
9. **Arbiters Can Only Rate Assigned Matches Once**: Preventing rating changes or rating unassigned matches typically requires triggers or application logic.

---

5. Conclusion

Our ER design for **ChessDB** captures the essential entities and relationships needed for managing chess tournaments. While we can depict key structures and basic constraints, more complex rules (e.g., time overlaps, match uniqueness, rating restrictions) must be handled via **SQL constraints**, **triggers**, or **application logic** once the design is mapped into a relational schema.