Umut Şendağ 2021400072
Ali Gökçek 2021400072

CMPE 321 Project 1 Part 2: Logical Database Design Report

1. Introduction

This second phase of **ChessDB** design maps our ER model to a **relational schema**. We define tables for users, players, coaches, arbiters, certificates, specialities, title, teams, sponsors, match tables, chess matches, halls, and tournaments, specifying **primary keys**, **foreign keys**, **CHECK** constraints, and **UNIQUE** constraints where possible. Some complex rules (e.g., overlapping time checks) remain only partially or not enforced due to the prohibition on triggers and procedural logic. Our code can be found at https://github.com/aligokcek1/Cmpe321_P1/blob/main/query.sql

2. Relational Model Overview

1. **USER(username PK)**: Basic user info (password, name, surname, nationality).
2. **PLAYER(username PK, FK→USER)**: Includes fide_ID, elo_rating (> 1000).
3. **COACH(username PK, FK→USER) & ARBITER(username PK, FK→USER)**: Additional attributes like experience_level for arbiters.
4. **TITLE(title_ID PK UNIQUE, title_name):** Title info for players.
5. **TEAM(team_ID PK)**: References COACH(username) (one coach per team) and SPONSOR(sponsor_ID).
6. **SPONSOR(sponsor_ID PK)**: Stores sponsor data.
7. **CERTIFICATE(certificate_type PK) & SPECIALITY(speciality_type PK):** Stores certificate and speciality data.
8. **TOURNAMENT(tournament_ID PK)**: References ARBITER.username as chief_arbiter.
9. **HALL(hall_ID PK)**: Holds hall info.
10. **MATCH_TABLE(table_ID PK, hall_ID FK→HALL)**: Tables belong to a specific hall.

11. **CHESS_MATCH(match_ID PK)**: References TOURNAMENT, ARBITER, HALL, MATCH_TABLE; includes time_slot (1–4), rating (1–10), white_player, black_player, etc. Possibly uses a UNIQUE(hall_ID, table_ID, date, time_slot) to minimize collisions.

---

3. Constraints Addressed

- **Primary Keys**: Guarantee entity identity in each table.
- **Foreign Keys**: Enforce referential integrity for relationships (e.g., matches cannot reference a nonexistent tournament).
- **CHECK Constraints**: For numeric limits (e.g., elo_rating>1000, rating in [1..10], time_slot in [1..4]).
- **UNIQUE**: Potential composite keys to avoid scheduling collisions in the same hall/table/time.

---

4. Discussion: Constraints Unrepresented or Partially Represented

Because triggers and procedural logic are prohibited, some constraints remain only partially enforced or not enforced by this relational schema:

1. **Time Overlap Checks:**
   - We can create UNIQUE (hall_ID, table_ID, date, time_slot) to block two matches with the same hall/table/date/time_slot from existing.
   - However, we cannot fully handle the scenario where a match spans two consecutive slots (e.g., time_slot 2 and 3). We also cannot stop an arbitrator or a player from being assigned to overlapping matches using only static DDL constraints.
2. **Coach Managing Only One Team at a Time:**
   - We rely on the TEAM (coach_username) design. A UNIQUE constraint on the coach_username field in TEAM will ensure that each coach is assigned to at most one team *in that table*.

- But if a coach can manage multiple teams sequentially (in different time periods), we cannot validate time intervals or transitions without triggers.

3. **Preventing Overlaps for Players and Arbiters:**
   - We cannot prevent a single player or arbiter from being scheduled in two matches at the same or overlapping time slots purely with standard DDL constraints. Checking for overlap is a multi-row temporal constraint, typically requiring triggers or application logic.
4. **Rating Cannot Be Changed Once Set:**
   - We cannot enforce "no updates after initial insert" via basic DDL. This behavior requires a trigger or application-level rules.
5. **Determining Tournament Winner:**
   - The logic to compare total match wins among teams cannot be baked into standard DDL. It would require queries or application-level code to compute an aggregate (no triggers or stored procedures allowed).

In summary, time conflicts, multi-tuple validations, and post-insert update prohibitions remain beyond the scope of standard SQL constraints alone. These features would require more advanced or procedural approaches that are currently not permitted.

---

5. Conclusion

Our **ChessDB** relational model uses **primary keys**, **foreign keys**, **UNIQUE**, and **CHECK** constraints where possible to enforce many structural and domain rules. However, certain essential scheduling, temporal, or multi-row constraints—like preventing overlapping matches or blocking rating changes—cannot be fully realized without **triggers**, **procedures**, or application logic. Despite these limitations, our schema captures the core relationships and entity constraints necessary for basic functionality and integrity of the tournament database.