CMPE 321 – Project 1: Logical Database Design Report (Part 2)

---

1. Introduction

This report presents the Logical Database Design for the Chess Tournament Database developed in Project 1 of CMPE 321. The logical design transforms the conceptual ER model into a relational schema using SQL Data Definition Language. This schema is implemented to support data integrity, normalization, and enforcement of domain-specific constraints related to users, tournaments, matches, and organizational structure.

---

2. Relational Schema

The following relational tables were created to implement the design. All primary keys, foreign keys, unique constraints, and checks have been defined in accordance with the requirements and normalized structure.

A. Core Tables

- **USER** (`username` PK, `password`, `name`, `surname`, `nationality`, `user_type`)
- **PLAYER** (`username` PK, `date_of_birth`, `elo_rating`, `fide_id`, `title_id` FK)
- **COACH** (`username` PK)
- **ARBITER** (`username` PK, `experience_level`)

B. Tournament and Team Tables

- **TEAM** (`team_id` PK, `team_name`, `coach_username` FK, `contract_start`, `contract_finish`, `sponsor_id` FK)
- **SPONSOR** (`sponsor_id` PK, `sponsor_name`)
- **TOURNAMENT** (`tournament_id` PK, `tournament_name`, `start_date`, `end_date`, `format`, `chief_arbiter_id` FK)

- **MATCH_** (match_id PK, tournament_id FK, match_date, time_slot, result, rating, white_player, black_player, white_player_team FK, black_player_team FK, assigned_arbiter FK, hall_id FK, table_id FK)

C. Supporting Tables

- **TITLE** (title_id PK, title_name)
- **HALL** (hall_id PK, hall_name, hall_country, hall_capacity)
- **CHESS_TABLE** (table_id PK, hall_id FK)
- **SPECIALTY** (username FK, specialty_name, PK(username, specialty_name))

D. Relationship Tables (Many-to-Many and Link Tables)

- **PLAYER_TEAM** (player_username FK, team_id FK, registration_date, PK(player_username, team_id))
- **TEAM_PLAYER** (team_id FK, player_username FK, registration_date, PK(team_id, player_username))
- **TOURNAMENT_TEAM** (tournament_id FK, team_id FK, registration_date, PK(tournament_id, team_id))
- **TOURNAMENT_HALL** (tournament_id FK, hall_id FK, PK(tournament_id, hall_id))
- **COACH_CERTIFICATE** (username FK, certificate_name, PK(username, certificate_name))
- **ARBITER_CERTIFICATE** (username FK, certificate_name, PK(username, certificate_name))

---

3. Integrity Constraints and Data Validation

To preserve data consistency and logical accuracy, the following constraints are enforced:

- **Primary Keys (PK):** Defined in every table to ensure unique identification.

- **Foreign Keys (FK):** Establish relationships across entities and enforce referential integrity.

- **Unique Constraints:** Enforced on critical attributes like sponsor_name, title_name, and fide_id.

- **Not Null Constraints:** Applied to all mandatory fields such as `username`, `team_name`, and `match_date`.

- **Check Constraints:**

    - `elo_rating` must be greater than 1000 in PLAYER.
    - `experience_level` in ARBITER must be one of ('Beginner', 'Intermediate', 'Advanced').
    - `time_slot` in MATCH_ must be between 1 and 4.
    - `white_player` ≠ `black_player`; `white_player_team` ≠ `black_player_team`.
- **Cascade / Set Null Policies:**

    - **ON DELETE CASCADE** for dependent records
    - **ON DELETE SET NULL** for optional fields

---

4. Constraints Not Represented in the Relational Schema

Some business rules could not be directly implemented in SQL DDL and must be handled using triggers, procedures, or at the application level:

- A player must only participate in tournaments where their team is registered.
- Arbiters can only rate the matches they are assigned to and cannot modify ratings.
- The team with the most match wins at the end of a tournament is declared the winner.
- The team with the most match wins at the end of the tournament is declared the winner.

---

5. Code Written in Mysql Workbench

```
-- Create the database if it doesn't exist
CREATE DATABASE IF NOT EXISTS cmpe_321;
-- Switch to this database
USE cmpe_321;
-- Tables that are not dependent on others
-- Table for chess titles like Grandmaster or FIDE Master
CREATE TABLE TITLE (
    title_id INT AUTO_INCREMENT PRIMARY KEY,
    title_name VARCHAR(255) NOT NULL UNIQUE
);
-- Table for sponsors
CREATE TABLE SPONSOR (
    sponsor_id INT AUTO_INCREMENT PRIMARY KEY,
    sponsor_name VARCHAR(255) NOT NULL UNIQUE
);
CREATE TABLE TEAM_PLAYER (
    team_id INT,
    player_username VARCHAR(255),
    registration_date DATE,
    PRIMARY KEY (team_id, player_username),
    FOREIGN KEY (team_id) REFERENCES TEAM(team_id) ON DELETE CASCADE,
    FOREIGN KEY (player_username) REFERENCES PLAYER(username) ON DELETE CASCADE
);
-- Table for coach specialties (like openings, tactics)
CREATE TABLE SPECIALTY (
    username VARCHAR(255),
    specialty_name VARCHAR(255),
    PRIMARY KEY (username, specialty_name),
    FOREIGN KEY (username) REFERENCES Coach(username) ON DELETE CASCADE
);
-- Table for physical halls where matches happen
CREATE TABLE HALL (
    hall_id INT AUTO_INCREMENT PRIMARY KEY,
    hall_name VARCHAR(255) NOT NULL,
    hall_country VARCHAR(255),
    hall_capacity INT
);
-- Table for chess tables in each hall
CREATE TABLE CHESS_TABLE (
    table_id INT AUTO_INCREMENT PRIMARY KEY,
    hall_id INT NOT NULL,
    FOREIGN KEY (hall_id) REFERENCES HALL(hall_id) ON DELETE CASCADE
);
-- Main user table (used by all user types)
CREATE TABLE USER (
    username VARCHAR(255) PRIMARY KEY,
    password VARCHAR(255) NOT NULL,
    name VARCHAR(255) NOT NULL,
    surname VARCHAR(255) NOT NULL,
    nationality VARCHAR(100) NOT NULL,
    user_type ENUM('PLAYER', 'COACH', 'ARBITER') NOT NULL
);
-- Table for coaches (subtype of USER)
CREATE TABLE COACH (
    username VARCHAR(255) PRIMARY KEY,
    FOREIGN KEY (username) REFERENCES USER(username) ON DELETE CASCADE
```

```
);
-- Table linking coaches to certificates (many-to-many)
CREATE TABLE COACH_CERTIFICATE (
    username VARCHAR(255),
    certificate_name VARCHAR(255),
    PRIMARY KEY (username, certificate_name),
    FOREIGN KEY (username) REFERENCES Coach(username) ON DELETE CASCADE
);
-- Table for arbiters (subtype of USER)
CREATE TABLE ARBITER (
    username VARCHAR(255)  PRIMARY KEY,
    experience_level ENUM('Beginner', 'Intermediate', 'Advanced') NOT NULL,
    FOREIGN KEY (username) REFERENCES USER(username) ON DELETE CASCADE
);
-- Table linking arbiters to certificates (many-to-many)
CREATE TABLE ARBITER_CERTIFICATE (
    username VARCHAR(255),
    certificate_name VARCHAR(255),
    PRIMARY KEY (username, certificate_name),
    FOREIGN KEY (username) REFERENCES Arbiter(username) ON DELETE CASCADE
);
-- Table for chess teams
CREATE TABLE TEAM (
    team_id INT AUTO_INCREMENT PRIMARY KEY,
    team_name VARCHAR(255) NOT NULL,
    coach_username VARCHAR(255) UNIQUE NOT NULL,
    contract_start DATE NOT NULL,
    contract_finish DATE NOT NULL,
    sponsor_id INT,
    FOREIGN KEY (coach_username) REFERENCES Coach(username) ON DELETE CASCADE,
    FOREIGN KEY (sponsor_id) REFERENCES SPONSOR(sponsor_id)
);
-- Table for players (subtype of USER)
CREATE TABLE PLAYER (
    username VARCHAR(255) PRIMARY KEY,
    date_of_birth DATE NOT NULL,
    elo_rating INT CHECK (elo_rating > 1000),
    fide_id INT,
    title_id INT,
    FOREIGN KEY (username) REFERENCES USER(username) ON DELETE CASCADE,
    FOREIGN KEY (title_id) REFERENCES TITLE(title_id)
);
-- Many-to-many relationship between players and teams
CREATE TABLE PLAYER_TEAM (
    player_username VARCHAR(255),
    team_id INT,
    registration_date DATE,
    PRIMARY KEY (player_username, team_id),
    FOREIGN KEY (player_username) REFERENCES PLAYER(username) ON DELETE CASCADE,
    FOREIGN KEY (team_id) REFERENCES TEAM(team_id) ON DELETE CASCADE
);
-- Table for chess tournaments
CREATE TABLE TOURNAMENT (
    tournament_id INT AUTO_INCREMENT PRIMARY KEY,
    tournament_name VARCHAR(255) NOT NULL,
    start_date DATE,
    end_date DATE,
    format VARCHAR(255),
    chief_arbiter_id INT,
    FOREIGN KEY (chief_arbiter_id) REFERENCES ARBITER(arbiter_id) ON DELETE SET NULL
);
-- Table to show which halls are used in each tournament
CREATE TABLE TOURNAMENT_HALL (
    tournament_id INT,
    hall_id INT,
    PRIMARY KEY (tournament_id, hall_id),
```

```
    FOREIGN KEY (tournament_id) REFERENCES TOURNAMENT(tournament_id) ON DELETE CASCADE,
    FOREIGN KEY (hall_id) REFERENCES HALL(hall_id) ON DELETE CASCADE
);
-- Table for team participation in tournaments
CREATE TABLE TOURNAMENT_TEAM (
    tournament_id INT,
    team_id INT,
    registration_date DATE,
    PRIMARY KEY (tournament_id, team_id),
    FOREIGN KEY (tournament_id) REFERENCES TOURNAMENT(tournament_id) ON DELETE CASCADE,
    FOREIGN KEY (team_id) REFERENCES TEAM(team_id) ON DELETE CASCADE
);
-- Table for matches played in tournaments
CREATE TABLE MATCH_ (
    match_id INT AUTO_INCREMENT PRIMARY KEY,
    tournament_id INT NOT NULL,
    hall_id INT NOT NULL,
    table_id INT NOT NULL,
    time_slot INT NOT NULL CHECK (time_slot BETWEEN 1 AND 4),
    match_date DATE NOT NULL,
    white_player_team INT NOT NULL,
    white_player VARCHAR(255) NOT NULL,
    black_player_team INT NOT NULL,
    black_player VARCHAR(255) NOT NULL,
    result ENUM('white_wins', 'black_wins', 'draw') NOT NULL,
    assigned_arbiter VARCHAR(255) NOT NULL,
    rating INT CHECK (rating BETWEEN 1 AND 10),
    UNIQUE (hall_id, table_id, match_date, time_slot),
    UNIQUE (match_date, time_slot, white_player),
    UNIQUE (match_date, time_slot, black_player),
    UNIQUE (match_date, time_slot, assigned_arbiter),
    FOREIGN KEY (tournament_id) REFERENCES TOURNAMENT(tournament_id) ON DELETE CASCADE,
    FOREIGN KEY (hall_id) REFERENCES HALL(hall_id),
    FOREIGN KEY (table_id) REFERENCES CHESS_TABLE(table_id),
    FOREIGN KEY (white_player_team) REFERENCES TEAM(team_id),
    FOREIGN KEY (white_player) REFERENCES PLAYER(username),
    FOREIGN KEY (black_player_team) REFERENCES TEAM(team_id),
    FOREIGN KEY (black_player) REFERENCES PLAYER(username),
    FOREIGN KEY (assigned_arbiter) REFERENCES ARBITER(username) ON DELETE CASCADE,
    CHECK (white_player != black_player),
    CHECK (white_player_team != black_player_team)
);
```

6. Conclusion

The logical database schema successfully translates the conceptual ER model into an implementable structure. It ensures normalized design, referential integrity, and rule enforcement through key constraints and checks. This schema forms a reliable foundation for database population, query execution, and application development in the next project phase.