PR 1

Database Design

Alejandro Pérez Bueno

Apr 22, 2025

Table of Contents

Exercise 1

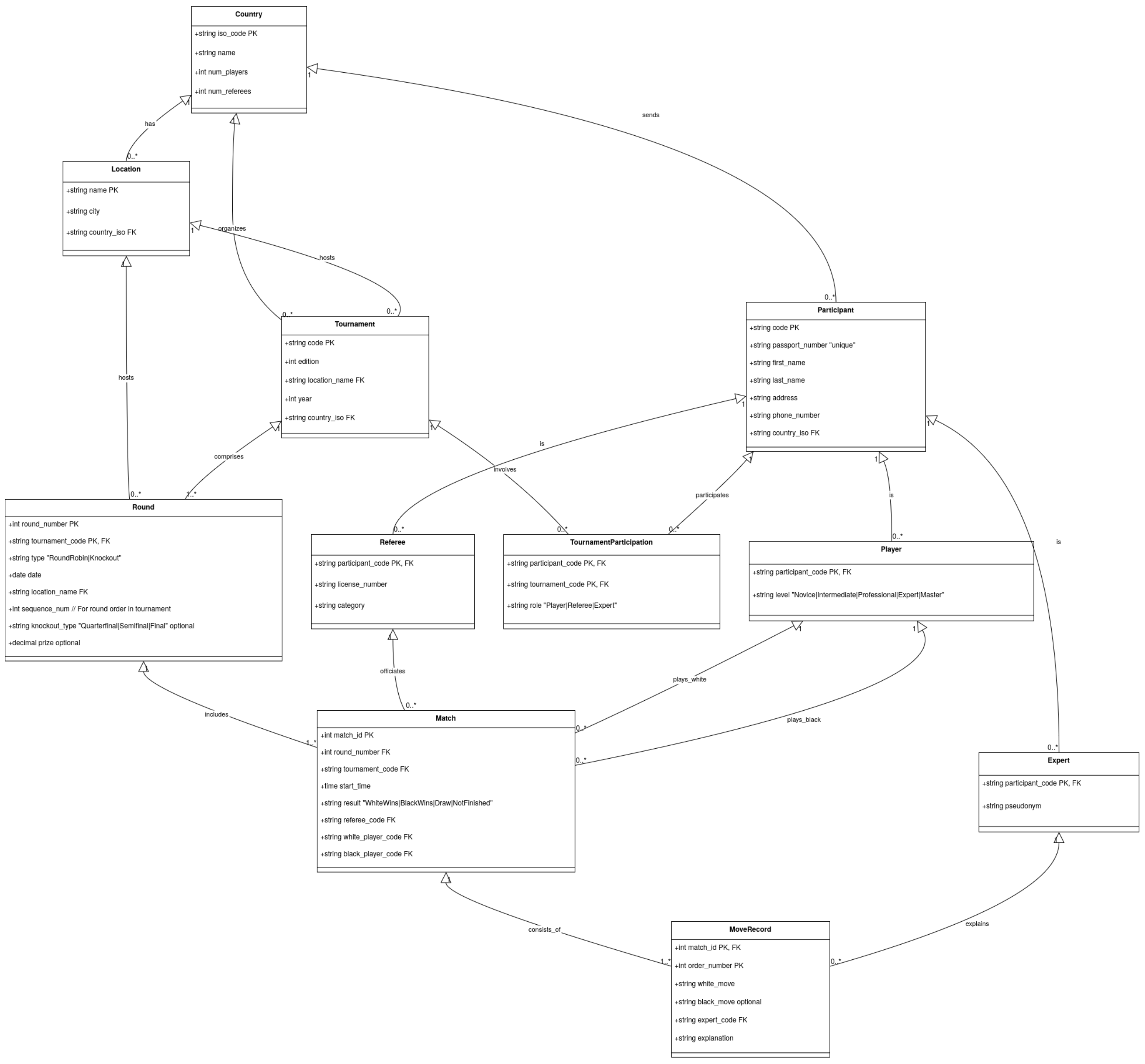

Figure 1: Diagram

Entities & Keys

- **Country:** ISO code PK, stores name, players, referees
- **Location:** PK per name, references country; only locations where tournaments are held are stored
- **Tournament:** Identified by code, edition, location, year, country (location and country as FK)
- **Participant:** code PK, also passport_number unique, references country. Participants can be players, referees, experts, or combinations as described.
- **Player, Referee, Expert:** Subclasses (PK = FK to Participant). Note that a participant can be player or referee or expert (ref & expert can overlap, player & ref cannot). Expert has optional pseudonym.

- **TournamentParticipation:** Associates participants, tournaments, and roles (disallows player/referee overlap; allows ref/expert).
- **Round:** PK = (tournament_code, round_number); can be round-robin (with round_number), or knockout (with knockout_type/prize if elimination round).
- **Match:** PK = match_id, with round/tournament FK, referee FK, white/black players.
- **MoveRecord:** PK = (match_id, order_number), move coordinates, explanation from expert.

Associations & Multiplicities

- Each country may have several locations (1:M)
- Each tournament in one location and country (1:M)
- Each participant from a country (can only represent one), and participate in multiple tournaments.
- Referees cannot play as players.
- Experts who are referees possible; experts and referees are not disjoint (player/referee cannot overlap).
- Rounds per tournament; matches per round; each match: exactly 2 players, 1 referee.
- Each move record in a match, explained by one expert (did not say if explanation is mandatory, but matches descriptions).
- Not all experts may participate in a season.

Constraints/Notes

- If `Participant` is a referee, CANNOT be a player (enforced by application or logic).
- Referee & expert overlap allowed (separate roles, same participant possible).
- A participant can only represent one country per season/tournament.
- Rounds: `type` field distinguishes between round-robin and knockout; if knockout, extra fields for stage/prize.
- A player can play only one match per round (implicit, enforced by logic/application).
- Each match: one referee, two players, distinct participants.
- Each round at a location; two rounds can be on the same date.
- Only locations used for tournaments stored.

Requirements / Constraints NOT EXPLICITLY MODELED or Ambiguous

- **Disjointness Player-Referee:** This cannot be enforced directly in UML/Mermaid, noted for implementers.
- **Round sequences & knockout types:** If needed, model could include additional context (e.g. which round is "quarterfinal", etc).
- **Unique phone/address/pseudonym, etc.:** Unless stated, not assumed unique.
- **Location as Name:** Assumed name uniquely identifies a place (may not be true in real world).
- **Tournament per Year:** No explicit (tournament, year) unique constraint, but possibly needed.
- **Prizes only for Knockout Rounds:** Model allows a prize only for knockout-type rounds.
- **Players per Country:** Countries may have zero participants; numbers stored as attributes, can be calculated from relations; could be derived.
- **Not all experts participate:** No explanation for unassigned move records, but allows Null expert if not used.

- **Move Explanations:** One expert per move record.
- **Address and phone optional for participants.**

Further Assumptions/Limitations

- **No intermediate tables for many-to-many** other than TournamentParticipation, explicitly.
- **Referee can arbitrarily many matches per round IF times don't overlap**; time overlaps not modeled at schema level, must be application-level.
- **Moves:** Each record always has white move, black move is usually present but not always (last if white delivers checkmate).
- **Cardinality indication via Mermaid syntax**; some semantic constraints must be enforced at the application level.

Exercise 2

1. Referee

Attribute	Type	NULL	Description
code	String	No	**PK**. Referee code
name	String	No	
nationality	String	No	
phone	String	Yes	
email	String	Yes	

- **PK:** code

2. Competition

Attribute	Type	NULL	Description
id	Integer	No	**PK**
name	String	No	**AK** (Assuming unique name)
fundationDate	Date	Yes	
type	String	No	Enum('league','cup','tournament')
address	String	Yes	
country	String	No	
competition_kind	String	No	'National'/'International'

- **PK:** id

- **AK:** name (if required)

3. WorksFor (Associative Table for Referee & Competition)

Attribute	Type	NULL	Description
refereeCode	String	No	**PK, FK** (Referee.code)
competitionId	Integer	No	**PK, FK** (Competition.id)
initDate	Date	Yes	

- **PK:** (refereeCode, competitionId)
- **FK:** refereeCode → Referee.code

- **FK:** competitionId → Competition.id

4. Game

Attribute	Type	NULL	Description
id	Integer	No	**PK**
date	Date	No	
result	String	Yes	
stadium	String	Yes	
competitionId	Integer	No	**FK** → Competition.id
hostingClubId	Integer	No	**FK** → Club.code
visitingClubId	Integer	No	**FK** → Club.code

- **PK:** id
- **FK:** competitionId → Competition.id

- **FK:** hostingClubId → Club.code

- **FK:** visitingClubId → Club.code

5. Club

Attribute	Type	NULL	Description
code	Integer	No	**PK**
name	String	No	**AK** (Assuming unique name)
address	String	Yes	
stadium	String	No	
competitionId	Integer	Yes	**FK** → Competition.id

- **PK:** code
- **AK:** name (if required)
- **FK:** competitionId → Competition.id
  - (Assumption, since Clubs "participate" in Competitions; adjust if Many-to-Many is needed.)

6. Player

Attribute	Type	NULL	Description
code	String	No	**PK**
name	String	No	
position	String	No	
salary	Float	Yes	
clubId	Integer	No	**FK** → Club.code
cityId	Integer	Yes	**FK** → City.code

- **PK:** code
- **FK:** clubId → Club.code
- **FK:** cityId → City.code

7. City

Attribute	Type	NULL	Description
code	Integer	No	**PK**
name	String	No	
province	String	Yes	
county	String	Yes	

- **PK:** code

Relationships (Associative Tables)

1. **WorksFor** ([see above])
2. **Player–Club**: One Player belongs to one Club.

3. **Player–City**: Player lives in a City.

4. **Competition–Club**: Many Clubs can participate in many Competitions.
   - If this is a true M:N, we'd need: `sql Table Competition_Club ( competitionId INTEGER -- FK to Competition.id clubCode INTEGER -- FK to Club.code PRIMARY KEY (competitionId, clubCode) )`
5. **Game–Club**: Hosting and visiting Club are captured as FKs in Game.

Summary Table

Table	Primary Keys	Alternate Keys	Foreign Keys (to)	Attributes that can be NULL
Referee	code			phone, email
Competition	id	name (optional)		fundationDate, address
WorksFor	refereeCode, competitionId		refereeCode→Referee.code, competitionId→Competition.id	initDate
Game	id		competitionId→Competition.id, hostingClubId & visitingClubId→Club.code	result, stadium
Club	code	name (optional)	competitionId→Competition.id (if applicable)	address
Player	code		clubId→Club.code, cityId→City.code	salary, cityId
City	code			province, county
Competition_Club	competitionId, clubCode		competitionId→Competition.id, clubCode→Club.code	

Enumerations

- Competition.type: {league, cup, tournament}
- Competition.competition_kind: {National, International}

Nullable Attributes

Attribute	May be NULL?	Notes
Referee.phone	Yes	Optional phone
Referee.email	Yes	Optional email
Competition.fundationDate	Yes	May be unknown
Competition.address	Yes	
WorksFor.initDate	Yes	May not always be set
Game.result	Yes	Game may not be played yet
Game.stadium	Yes	May be different/stadium not set
Club.address	Yes	Optional
Player.salary	Yes	Possibly unknown
Player.cityId	Yes	Player may not have registered residence
City.province	Yes	Depending on country
City.county	Yes	

Exercise 3

a) Normal Form Analysis and BCNF Changes

Given relations:

- **Championship(championshipID, name, startDate, location)**
- **Player(playerID, name, rating, championshipsPlayed)**
- **Game(gameID, championshipID, date, whitePlayerID, blackPlayerID)**

Foreign Keys: - championshipID in Game → Championship - whitePlayerID, blackPlayerID in Game → Player

What Normal Form are these in? Justify.

1NF:
All "attributes" are atomic (dates, integers, names, etc.), so relations are in 1NF.

2NF:
- No partial dependency on a part of a composite key (because all primary keys are simple except in Game, whose PK is gameID). - Each non-key attribute in each table depends on the entire PK. - So, all are in 2NF.

3NF:
- **Championship**: All non-key attributes depend on championshipID. - **Player**: All attributes depend only on playerID. - **Game**: All attributes depend only on gameID. - No transitive dependencies. - Therefore, all relations are in 3NF.

BCNF:
- In all tables, every determinant is a candidate key.
- No non-trivial FDs where the determinant is not a superkey.

All relations are already in BCNF. No changes needed.

b) Add Referee License Number and Name: Analyze Normal Form, BCNF Changes

- **New attributes in Game table: refereeLicenseNum, refereeName**

So, Game(gameID, championshipID, date, whitePlayerID, blackPlayerID, refereeLicenseNum, refereeName)

Normal Form Now

Assume: - A referee may referee multiple games. - A referee's license number uniquely identifies the referee. - refereeLicenseNum determines refereeName (FD: refereeLicenseNum → refereeName).

First, 1NF: Verified. 2NF: The PK is gameID (a simple key) – all non-key attributes depend on it.

3NF: - refereeLicenseNum determines refereeName - But refereeName is not determined by the PK, but by refereeLicenseNum. - There is a transitive dependency: gameID → refereeLicenseNum → refereeName

Thus, **Game is NOT in 3NF nor BCNF**.

BCNF Decomposition

Decompose into:

- **Game**: (gameID, championshipID, date, whitePlayerID, blackPlayerID, refereeLicenseNum)
- **Referee**: (refereeLicenseNum, refereeName)

Where: - **Game.refereeLicenseNum** is a FK to Referee.refereeLicenseNum.

Both are now in BCNF.

c) Add Referee Categories

Requirement: Each referee can participate in multiple categories (and vice versa).

- This is a many-to-many relationship: Referees   Categories

Normal Form Impact

You'll need:

- **Category(categoryID, categoryName, ...)**
- **RefereeCategory(refereeLicenseNum, categoryID)**

Assume: - refereeLicenseNum is PK in Referee. - categoryID is PK in Category. - refereeLicenseNum + categoryID composite PK in RefereeCategory.

Are these relations in BCNF?

- Each table's candidate keys are the determinants of their FDs; thus, all are in BCNF (assuming no partial or transitive dependencies in the new tables).

d) Player Pairs' First Match Info: To BCNF

Requirement: Know, for each pair of players, the year they first faced each other, and the championship name.

Attributes needed:
(player1Name, player2Name, year, championshipName)

How should you design this?

Since names are not necessarily unique (should use playerID), but if names uniquely identify players in this context, then:

A possible relation:

- **PlayerPairFirstMatch(player1ID, player2ID, year, championshipID)**

where: - (player1ID, player2ID) are IDs of players such that player1ID < player2ID (to avoid duplicate pairs) - year is integer, championshipID is FK to Championship

If you include names directly: - **(player1ID, player2ID, year, championshipID, player1Name, player2Name, championshipName)** - But player1Name depends on player1ID, and championshipName on championshipID, so this will not be in 3NF.

To achieve BCNF:
- Use only IDs in the relation: (player1ID, player2ID, year, championshipID) - Fetch player names and championship name via join if needed.