Database Schema Analysis and Normalization

Halil Karabacak 2020400186 - Ceyhun Sonyürek 2020400258

Database Schema Analysis

Each table's SQL definition is followed by its analysis regarding functional dependencies and normalization forms.

USER Table

SQL Definition:

```
CREATE TABLE USER (
    username VARCHAR(50),
    name VARCHAR(50),
    surname VARCHAR(50),
    password VARCHAR(50),
    PRIMARY KEY (username)
);
```

Functional Dependencies:

- username → name, surname, password

Normal Form Analysis:

- This table is in **Boyce-Codd Normal Form (BCNF)** because all non-trivial functional dependencies have the primary key on the left-hand side.

COACH Table

SQL Definition:

```
CREATE TABLE COACH (
    username VARCHAR(50),
    nationality VARCHAR(50) NOT NULL,
    PRIMARY KEY (username),
    FOREIGN KEY (username) REFERENCES USER (username)
);
```

Functional Dependencies:

- username → nationality

Normal Form Analysis:

- This table is in BCNF. The only functional dependency uses the primary key.

Players Table

SQL Definition:

```
CREATE TABLE Players (
    username VARCHAR(50),
    date_of_birth DATE,
    weight DECIMAL(5,2),
    height DECIMAL(5,2),
    PRIMARY KEY (username),
    FOREIGN KEY (username) REFERENCES USER (username)
);
```

Functional Dependencies:

- username → date_of_birth, weight, height

Normal Form Analysis:

- This table is in BCNF as the left-hand side of the functional dependency is the primary key.

Juries Table

SQL Definition:

```
CREATE TABLE Juries (
    username VARCHAR(50),
    nationality VARCHAR(50),
    PRIMARY KEY (username),
    FOREIGN KEY (username) REFERENCES USER (username)
);
```

Functional Dependencies:

- username → nationality

Normal Form Analysis:

- The table is in BCNF as the primary key is the only determinant.

Positions Table

SQL Definition:

```
CREATE TABLE Positions (
    position_id INT,
    position_name VARCHAR(50) UNIQUE,
    PRIMARY KEY (position_id)
);
```

Functional Dependencies:

- position_id → position_name
- position_name → position_id

Normal Form Analysis:

- This table is in BCNF as both attributes function as keys.

TV_CHANNEL Table

SQL Definition:

```
CREATE TABLE TV_CHANNEL (
    channel_ID INT,
    channel_name VARCHAR(100) NOT NULL UNIQUE,
    PRIMARY KEY (channel_ID)
);
```

Functional Dependencies:

- channel_ID → channel_name

Normal Form Analysis:

- This table is in BCNF as both attributes can serve as keys.

Teams Table

SQL Definition:

```
CREATE TABLE Teams (
    team_ID INT,
    team_name VARCHAR(100),
    coach_username VARCHAR(50) NOT NULL UNIQUE,
    contract_start DATE,
    contract_finish DATE,
    channel_ID INT NOT NULL,
    PRIMARY KEY (team_ID),
    FOREIGN KEY (coach_username) REFERENCES COACH (username),
    FOREIGN KEY (channel_ID) REFERENCES TV_CHANNEL (channel_ID)
);
```

Functional Dependencies:

- team_ID → team_name, coach_username, contract_start, contract_finish, channel_ID
- coach_username → team_ID

Normal Form Analysis:

- This table is not in BCNF because the non-primary key *coach_username* determines *team_ID*.
- **Normalization:** Split into two tables, one for *Teams* (team_ID, team_name, contract_start, contract_finish, channel_ID) and another for *Team_Coach* (coach_username, team_ID) to maintain BCNF.

Player_Position Table

SQL Definition:

```
CREATE TABLE Player_Position (
    player_username VARCHAR(50),
    position_ID INT,
    PRIMARY KEY (player_username, position_ID),
    FOREIGN KEY (player_username) REFERENCES Players (username),
    FOREIGN KEY (position_ID) REFERENCES Positions (position_id)
);
```

Functional Dependencies:

- (player_username, position_ID) → None

Normal Form Analysis:

- This table is in BCNF because it has a composite primary key and no other attributes.

Stadium Table

SQL Definition:

```
CREATE TABLE Stadium (
    stadium_ID INT,
    stadium_name VARCHAR(100) NOT NULL UNIQUE,
    stadium_country VARCHAR(50),
    PRIMARY KEY (stadium_ID)
);
```

Functional Dependencies:

- stadium_ID → stadium_name, stadium_country

Normal Form Analysis:

- This table is in BCNF because the primary key is the only determinant.

Match_Session Table

SQL Definition:

```
CREATE TABLE Match_Session (
    session_ID INT NOT NULL UNIQUE,
    stadium_ID INT NOT NULL,
    team_ID INT NOT NULL,
    date DATE,
    time_slot INT CHECK (time_slot >0 AND time_slot <5),
    assigned_jury_username VARCHAR(50) NOT NULL,
    UNIQUE (stadium_ID, date, time_slot),
    FOREIGN KEY (stadium_ID) REFERENCES Stadium (stadium_ID),
    FOREIGN KEY (team_ID) REFERENCES Teams (team_ID),
    FOREIGN KEY (assigned_jury_username) REFERENCES Juries (username),
    PRIMARY KEY (session_ID)
);
```

Functional Dependencies:

- session_ID → stadium_ID, team_ID, date, time_slot, assigned_jury_username
- (stadium_ID, date, time_slot) → session_ID

Normal Form Analysis:

- This table is in BCNF as `session_ID` and the composite of (`stadium_ID`, `date`, `time_slot`) are superkeys.

Player_Team Table

SQL Definition:

```
CREATE TABLE Player_Team (
    team_ID INT,
    player_username VARCHAR(50),
    PRIMARY KEY (team_ID, player_username),
    FOREIGN KEY (team_ID) REFERENCES Teams (team_ID),
    FOREIGN KEY (player_username) REFERENCES Players (username)
);
```

Functional Dependencies:

- (`team_ID`, `player_username`) → None

Normal Form Analysis:

- This table is in BCNF because it only contains the composite primary key.

Player_Match_Position Table

SQL Definition:

```
CREATE TABLE Player_Match_Position (
    player_username VARCHAR(255),
    session_ID INT,
    position_ID INT,
    PRIMARY KEY (player_username, session_ID),
    FOREIGN KEY (player_username) REFERENCES Players (username),
    FOREIGN KEY (position_ID) REFERENCES Positions (position_id),
    FOREIGN KEY (session_ID) REFERENCES Match_Session (session_ID)
);
```

Functional Dependencies:

- (`player_username`, `session_ID`) → `position_ID`

Normal Form Analysis:

- This table is not in BCNF as `position_ID` is determined by the composite key. However, this arrangement is practical for ensuring that position assignments are clearly defined for each match and player combination.

Players_in_Session Table

SQL Definition:

```
CREATE TABLE Players_in_Session (
    player_username VARCHAR(50),
    session_ID INT,
    PRIMARY KEY (player_username, session_ID),
    FOREIGN KEY (player_username) REFERENCES Players (username),
    FOREIGN KEY (session_ID) REFERENCES Match_Session (session_ID)
);
```

Functional Dependencies:

- (`player_username`, `session_ID`) → None

Normal Form Analysis:

- This table is in BCNF because it only contains the composite primary key.

Rating_Relationship Table

SQL Definition:

```
CREATE TABLE Rating_Relationship (
    rating DECIMAL(2,1),
    session_ID INT,
    assigned_jury_username VARCHAR(50) NOT NULL,
    PRIMARY KEY (session_ID),
    FOREIGN KEY (session_ID) REFERENCES Match_Session (session_ID),
    FOREIGN KEY (assigned_jury_username) REFERENCES Juries (username)
);
```

Functional Dependencies:

- session_ID $\rightarrow$ rating, assigned_jury_username

Normal Form Analysis:

- This table is in BCNF.

Database Integrity Triggers

Avoiding Overlapping Sessions in the Same Stadium

This trigger ensures that no two match sessions overlap in the same stadium at the same time. It is a measure that acts before a new record is inserted into the `MatchSession` table. If an overlap is detected, it aborts the operation by raising a signal error.

```
DELIMITER $$

CREATE TRIGGER CheckSessionConflict
BEFORE INSERT ON MatchSession
FOR EACH ROW
BEGIN
    DECLARE conflict_count INT;

    -- Count existing sessions that match the criteria
    SELECT COUNT(*) INTO conflict_count FROM MatchSession
    WHERE stadium_ID = NEW.stadium_ID
      AND date = NEW.date
      AND ABS(time_slot = NEW.time_slot) <= 1;

    -- If any existing sessions are found, prevent the insert
    IF conflict_count > 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Conflict␣detected␣for␣
            the␣requested␣time␣and␣stadium.';
    END IF;
END$$

DELIMITER ;
```

Preventing Player Overlaps Across Sessions

This trigger prevents players from being scheduled in overlapping match sessions. It checks for time conflicts before inserting a new record into the `SessionSquads` table. If a conflict is identified, the insertion is blocked.

```
DELIMITER //

CREATE TRIGGER prevent_time_conflict
BEFORE INSERT ON SessionSquads
FOR EACH ROW
BEGIN
    DECLARE conflict_count INT;

    -- Check for any existing match sessions assigned to the player that
       overlap with the new match session time
    SELECT COUNT(*) INTO conflict_count
    FROM SessionSquads sq
    JOIN MatchSession ms ON sq.session_ID = ms.session_ID
    JOIN MatchSession new_ms ON new_ms.session_ID = NEW.session_ID
    WHERE sq.played_player_username = NEW.played_player_username
      AND new_ms.date = ms.date
      AND ABS(new_ms.time_slot = ms.time_slot) <= 1;

    -- If there is a time conflict, prevent the insert operation
    IF conflict_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot␣assign␣player␣to␣multiple␣matches␣at␣the
           ␣same␣time␣slot.';
    END IF;
END;

//
DELIMITER ;
```