Chức năng:

* Cầu thủ

Cầu thủ ghi nhiều bàn nhất (vua phá lưới)

CREATE VIEW TopScorers AS

SELECT

P.Player\_id,

P.Name AS Player\_Name,

P.Team\_id,

T.Name AS Team\_Name,

SUM(MP.Goals) AS Total\_Goals

FROM

Player P

JOIN

Match\_Player MP ON P.Player\_id = MP.Player\_id

JOIN

Team T ON P.Team\_id = T.Team\_id

GROUP BY

P.Player\_id, P.Name, P.Team\_id, T.Name

ORDER BY

Total\_Goals DESC;

Cầu thủ lương cao nhất

CREATE VIEW TopPaidPlayersRanking AS

SELECT

Player\_id,

Name AS Player\_Name,

Salary

FROM

Player

ORDER BY

Salary DESC;

* Đội

BXH hiển thị các thông tin gồm điểm, hiệu số bàn thắng thua, số bàn thắng, số trận đã đấu. Sort theo thứ tự này luôn (trừ số trận đã đấu)

CREATE VIEW bxh1 AS

SELECT

T.Team\_id,

T.Name,

COALESCE(MD.Matches, 0) AS Matches,

COALESCE(MD.Wins, 0) AS Wins,

COALESCE(MD.Draws, 0) AS Draws,

COALESCE(MD.Losses, 0) AS Losses,

COALESCE(MD.Goals\_For, 0) AS Goals\_For,

COALESCE(MD.Goals\_Against, 0) AS Goals\_Against,

COALESCE(MD.DIFF, 0) AS DIFF,

COALESCE(MD.Points, 0) AS Points

FROM

Team T

LEFT JOIN LATERAL

(SELECT

COUNT(CASE WHEN TJ.Home\_team\_id = T.Team\_id AND TJ.Home\_goals > TJ.Away\_goals THEN 1

WHEN TJ.Away\_team\_id = T.Team\_id AND TJ.Away\_goals > TJ.Home\_goals THEN 1

END) AS Wins,

COUNT(CASE WHEN TJ.Home\_goals = TJ.Away\_goals THEN 1 END) AS Draws,

COUNT(CASE WHEN TJ.Home\_team\_id = T.Team\_id AND TJ.Home\_goals < TJ.Away\_goals THEN 1

WHEN TJ.Away\_team\_id = T.Team\_id AND TJ.Away\_goals < TJ.Home\_goals THEN 1

END) AS Losses,

SUM(CASE WHEN TJ.Home\_team\_id = T.Team\_id THEN TJ.Home\_goals

WHEN TJ.Away\_team\_id = T.Team\_id THEN TJ.Away\_goals END) AS Goals\_For,

SUM(CASE WHEN TJ.Home\_team\_id = T.Team\_id THEN TJ.Away\_goals

WHEN TJ.Away\_team\_id = T.Team\_id THEN TJ.Home\_goals END) AS Goals\_Against,

SUM(CASE WHEN TJ.Home\_team\_id = T.Team\_id THEN TJ.Home\_goals

WHEN TJ.Away\_team\_id = T.Team\_id THEN TJ.Away\_goals END) -

SUM(CASE WHEN TJ.Home\_team\_id = T.Team\_id THEN TJ.Away\_goals

WHEN TJ.Away\_team\_id = T.Team\_id THEN TJ.Home\_goals END) AS DIFF,

(3 \* COUNT(CASE WHEN TJ.Home\_team\_id = T.Team\_id AND TJ.Home\_goals > TJ.Away\_goals THEN 1

WHEN TJ.Away\_team\_id = T.Team\_id AND TJ.Away\_goals > TJ.Home\_goals THEN 1

END) + COUNT(CASE WHEN TJ.Home\_goals = TJ.Away\_goals THEN 1 END)) AS Points,

COUNT(\*) AS Matches

FROM

TEAM\_JOIN TJ

WHERE

TJ.Home\_team\_id = T.Team\_id OR TJ.Away\_team\_id = T.Team\_id

GROUP BY

T.Team\_id) MD ON TRUE

ORDER BY

Points DESC, DIFF DESC, Goals\_For DESC, Name ASC;

* Trận đấu

Trận đấu có tỷ số đậm nhất (hiệu số bàn thắng 2 đội ghi được lớn nhất)

CREATE VIEW MaxGoalDifferenceMatch AS

SELECT

Match\_id,

Home\_team\_id,

Away\_team\_id,

Home\_goals - Away\_goals AS Goal\_Difference

FROM

TEAM\_JOIN

ORDER BY

Goal\_Difference DESC

LIMIT 1;

Trận đấu có nhiều thẻ vàng, đỏ nhất

Input:

* 2 đội (nhà, khách)
* tỉ số
* cầu thủ đội khách
* cầu thủ đội nhà, số phút thi đấu, kiến tạo, ghi bàn, thẻ vàng, thẻ đỏ
* check 2 đội đã gặp nhau chưa (đội 1 là home, đội 2 away thì check ko đc lặp lại nthe)
* check cầu thủ 2 đội có chấn thương, cấm thi đấu ko
* nhập cầu thủ chấn thương
* Khi input cầu thủ còn phải check số bàn bằng tổng số bàn lúc input trận đấu

Tạo trigger kiểm tra xem nhập đúng dữ liệu vào bảng match\_player hay chưa: - Đúng rồi

CREATE OR REPLACE FUNCTION check\_player\_eligibility()

RETURNS TRIGGER AS $$

DECLARE

is\_injured BOOLEAN;

has\_red\_card BOOLEAN=0;

match\_id\_in\_team BOOLEAN;

match\_red\_card INT;

BEGIN

-- Kiểm tra xem cầu thủ có chấn thương không

SELECT EXISTS (

SELECT 1

FROM Player\_Injury PI

WHERE PI.Player\_id = NEW.Player\_id

AND PI.Expected\_return >= CURRENT\_DATE

) INTO is\_injured;

-- Kiểm tra xem cầu thủ đã nhận thẻ đỏ ở trận trước không

SELECT 1, MP.match\_id

FROM Match\_Player MP

WHERE MP.Player\_id = NEW.Player\_id

AND MP.Red\_cards > 0

AND CEIL((NEW.Match\_id/10.0))-CEIL((MP.Match\_id/10.0)) = 1

INTO has\_red\_card,match\_red\_card;

SELECT EXISTS(

SELECT TJ.Match\_id

FROM TEAM\_JOIN TJ JOIN Player Pl ON(TJ.Home\_team\_id=Pl.team\_id OR TJ.Away\_team\_id=Pl.Team\_id)

WHERE NEW.Match\_id = TJ.Match\_id AND New.Player\_id=Pl.Player\_id)

INTO match\_id\_in\_team;

IF is\_injured THEN

RAISE EXCEPTION 'Player % is not allowed to play due to injury', NEW.Player\_id;

END IF;

IF has\_red\_card THEN

RAISE EXCEPTION 'Player % is not allowed to play due to red card received in match %', NEW.Player\_id, match\_red\_card;

END IF;

IF match\_id\_in\_team = False THEN

RAISE EXCEPTION 'Player % is not in this match %', NEW.Player\_id,NEW.Match\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Tạo trigger

CREATE OR REPLACE TRIGGER check\_player\_eligibility\_trigger

BEFORE INSERT ON Match\_Player

FOR EACH ROW

EXECUTE FUNCTION check\_player\_eligibility();

Tạo trigger kiểm tra không có trận đấu nào hai đội trùng nhau - đúng r

CREATE OR REPLACE FUNCTION check\_match\_limit()

RETURNS TRIGGER AS $$

DECLARE

meetings INT;

BEGIN

SELECT COUNT(\*) INTO meetings FROM TEAM\_JOIN

WHERE (Home\_team\_id = NEW.Home\_team\_id AND Away\_team\_id = NEW.Away\_team\_id);

IF meetings >= 1 THEN

RAISE EXCEPTION 'Each team can only meet twice, home and away.';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check\_match\_limit

BEFORE INSERT ON TEAM\_JOIN

FOR EACH ROW

EXECUTE FUNCTION check\_match\_limit();

Tạo trigger khi insert vào bảng match: trong một vòng thì mỗi trọng tài chỉ bắt 1 trận. – Đúng rồi

CREATE OR REPLACE FUNCTION EnsureOneMatchPerRefereePerRound()

RETURNS TRIGGER AS $$

DECLARE

count\_referee\_matches INT;

BEGIN

-- Kiểm tra xem trọng tài đã được chỉ định cho một trận đấu trong cùng một vòng chưa

SELECT COUNT(\*)

INTO count\_referee\_matches

FROM Match

WHERE Referee\_main\_id = NEW.Referee\_main\_id

AND CEIL(NEW.match\_id/10.0)=CEIL(match\_id/10.0);

-- Nếu trọng tài đã được chỉ định cho một trận đấu khác trong cùng một vòng, ngăn chặn việc chèn

IF count\_referee\_matches > 0 THEN

RAISE EXCEPTION 'Referee % can only officiate one match in round %.',NEW.referee\_main\_id,CEIL(NEW.match\_id/10.0);

END IF;

-- Trả về NEW nếu không có lỗi

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Tạo trigger

CREATE TRIGGER EnsureOneMatchPerRefereePerRound

BEFORE INSERT ON Match

FOR EACH ROW

EXECUTE FUNCTION EnsureOneMatchPerRefereePerRound();

Standings table can be generated at any time, even before the completion of round.

tạo function insert vào bảng match - đúng r

CREATE OR REPLACE FUNCTION InsertMatchesBatch(match\_data TEXT)

RETURNS VARCHAR(255) AS $$

DECLARE

i INT := 1;

single\_match\_data TEXT;

match\_id INT;

match\_date DATE;

match\_time TIME;

referee\_main INT;

referee\_control1 INT;

referee\_control2 INT;

referee\_control3 INT;

BEGIN

-- Loop through the match data to process each match

WHILE TRUE LOOP

-- Extract the data for a single match

single\_match\_data := SPLIT\_PART(match\_data, ';', i);

-- Check for the end of the input

IF single\_match\_data = '' THEN

RETURN CONCAT('Inserted ', i - 1, ' matches successfully');

END IF;

-- Extract individual fields from the match data

match\_id = CAST(SPLIT\_PART(single\_match\_data, ',', 1) AS INT);

match\_date = TO\_DATE(SPLIT\_PART(single\_match\_data, ',', 2), 'YYYY-MM-DD');

match\_time = TO\_TIMESTAMP(SPLIT\_PART(single\_match\_data, ',', 3), 'HH24:MI:SS');

referee\_main = CAST(SPLIT\_PART(single\_match\_data, ',', 4) AS INT);

referee\_control1 = CAST(SPLIT\_PART(single\_match\_data, ',', 5) AS INT);

referee\_control2 = CAST(SPLIT\_PART(single\_match\_data, ',', 6) AS INT);

referee\_control3 = CAST(SPLIT\_PART(single\_match\_data, ',', 7) AS INT);

-- Insert the extracted data into the Match table

INSERT INTO Match (Match\_id, Date\_of\_match, Match\_time, Referee\_main\_id, Referee\_control1\_id, Referee\_control2\_id, Referee\_control3\_id)

VALUES (match\_id, match\_date, match\_time, referee\_main, referee\_control1, referee\_control2, referee\_control3);

i = i + 1;

END LOOP;

END;

$$ LANGUAGE plpgsql;

tạo function insert vào bảng team\_join - đúng r

CREATE OR REPLACE FUNCTION InsertTeamJoinBatch(team\_join\_data TEXT)

RETURNS VARCHAR(255) AS $$

DECLARE

i INT = 1;

single\_team\_join\_data TEXT;

match\_id INT;

home\_team\_id INT;

away\_team\_id INT;

home\_goals INT;

away\_goals INT;

BEGIN

-- Loop through the team join data to process each match

WHILE TRUE LOOP

-- Extract the data for a single match

single\_team\_join\_data = SPLIT\_PART(team\_join\_data, ';', i);

IF single\_team\_join\_data='' THEN

RETURN CONCAT('Inserted ', i - 1, ' team\_join successfully');

END IF;

-- Extract individual fields from the team join data

match\_id = CAST(SPLIT\_PART(single\_team\_join\_data, ',', 1) AS INT);

home\_team\_id = CAST(SPLIT\_PART(single\_team\_join\_data, ',', 2) AS INT);

away\_team\_id = CAST(SPLIT\_PART(single\_team\_join\_data, ',', 3) AS INT);

home\_goals = CAST(SPLIT\_PART(single\_team\_join\_data, ',', 4) AS INT);

away\_goals = CAST(SPLIT\_PART(single\_team\_join\_data, ',', 5) AS INT);

-- Insert the extracted data into the TEAM\_JOIN table

INSERT INTO TEAM\_JOIN (Match\_id, Home\_team\_id, Away\_team\_id, Home\_goals, Away\_goals)

VALUES (match\_id, home\_team\_id, away\_team\_id, home\_goals, away\_goals);

i = i + 1;

END LOOP;

END;

$$ LANGUAGE plpgsql;

tạo function insert vào bảng match\_player   
CREATE OR REPLACE FUNCTION Insert\_Match\_Player(match\_player\_data TEXT)

RETURNS VARCHAR(255) AS $$

DECLARE

i INT =1;

single\_match\_data TEXT;

p\_Player\_id INT;

p\_Match\_id INT;

p\_Minutes\_played INT;

p\_Goals INT;

p\_Assists INT;

p\_Yellow\_cards INT;

p\_Red\_cards INT;

BEGIN

WHILE TRUE LOOP

single\_match\_data=SPLIT\_PART(match\_player\_data,';',i);

IF single\_match\_data='' THEN

RETURN CONCAT('Inserted ',i-1,' successfully');

END IF;

p\_Player\_id=CAST(SPLIT\_PART(single\_match\_data,',',1) AS INT);

p\_Match\_id=CAST(SPLIT\_PART(single\_match\_data,',',2) AS INT);

p\_Minutes\_played=CAST(SPLIT\_PART(single\_match\_data,',',3) AS INT);

p\_Goals=CAST(SPLIT\_PART(single\_match\_data,',',4) AS INT);

p\_Assists=CAST(SPLIT\_PART(single\_match\_data,',',5) AS INT);

p\_Yellow\_cards=CAST(SPLIT\_PART(single\_match\_data,',',6) AS INT);

p\_Red\_cards=CAST(SPLIT\_PART(single\_match\_data,',',7) AS INT);

INSERT INTO Match\_Player(Player\_id, Match\_id, Minutes\_played, Goals, Assists, Yellow\_cards, Red\_cards)

VALUES (p\_Player\_id, p\_Match\_id, p\_Minutes\_played, p\_Goals, p\_Assists, p\_Yellow\_cards, p\_Red\_cards);

END LOOP;

END ;

$$ LANGUAGE plpgsql;

tạo function insert vào bảng player\_injury

CREATE OR REPLACE FUNCTION Insert\_Player\_Injury(player\_injury\_data TEXT)

RETURNS VARCHAR(255) AS $$

DECLARE

i INT = 1;

single\_injury\_data TEXT;

p\_Player\_id INT;

p\_Injury\_id INT;

p\_Injury\_date DATE;

p\_Expected\_return DATE;

p\_Injury\_notes VARCHAR(100);

BEGIN

WHILE TRUE LOOP

single\_injury\_data = SPLIT\_PART(player\_injury\_data, ';', i);

IF single\_injury\_data = '' THEN

RETURN CONCAT('Inserted ', i - 1, ' player injuries successfully');

END IF;

p\_Player\_id = CAST(SPLIT\_PART(single\_injury\_data, ',', 1) AS INT);

p\_Injury\_id = CAST(SPLIT\_PART(single\_injury\_data, ',', 2) AS INT);

p\_Injury\_date = TO\_DATE(SPLIT\_PART(single\_injury\_data, ',', 3), 'YYYY-MM-DD');

p\_Expected\_return = TO\_DATE(SPLIT\_PART(single\_injury\_data, ',', 4), 'YYYY-MM-DD');

p\_Injury\_notes = SPLIT\_PART(single\_injury\_data, ',', 5);

INSERT INTO Player\_Injury(Player\_id, Injury\_id, Injury\_date, Expected\_return, Injury\_notes)

VALUES (p\_Player\_id, p\_Injury\_id, p\_Injury\_date, p\_Expected\_return, p\_Injury\_notes);

i = i + 1;

END LOOP;

END;

$$ LANGUAGE plpgsql;

Tạo trigger kiểm tra xem nhập đúng dữ liệu vào bảng match\_player hay chưa: - Đúng rồi

CREATE OR REPLACE FUNCTION check\_player\_eligibility()

RETURNS TRIGGER AS $$

DECLARE

is\_injured BOOLEAN;

has\_red\_card BOOLEAN=0;

match\_id\_in\_team BOOLEAN;

match\_red\_card INT;

BEGIN

-- Kiểm tra xem cầu thủ có chấn thương không

SELECT EXISTS (

SELECT 1

FROM Player\_Injury PI

WHERE PI.Player\_id = NEW.Player\_id

AND PI.Expected\_return >= CURRENT\_DATE

) INTO is\_injured;

-- Kiểm tra xem cầu thủ đã nhận thẻ đỏ ở trận trước không

SELECT 1, MP.match\_id

FROM Match\_Player MP

WHERE MP.Player\_id = NEW.Player\_id

AND MP.Red\_cards > 0

AND CEIL((NEW.Match\_id/10.0))-CEIL((MP.Match\_id/10.0)) = 1

INTO has\_red\_card,match\_red\_card;

SELECT EXISTS(

SELECT TJ.Match\_id

FROM TEAM\_JOIN TJ JOIN Player Pl ON(TJ.Home\_team\_id=Pl.team\_id OR TJ.Away\_team\_id=Pl.Team\_id)

WHERE NEW.Match\_id = TJ.Match\_id AND New.Player\_id=Pl.Player\_id)

INTO match\_id\_in\_team;

IF is\_injured THEN

RAISE EXCEPTION 'Player % is not allowed to play due to injury', NEW.Player\_id;

END IF;

IF has\_red\_card THEN

RAISE EXCEPTION 'Player % is not allowed to play due to red card received in match %', NEW.Player\_id, match\_red\_card;

END IF;

IF match\_id\_in\_team = False THEN

RAISE EXCEPTION 'Player % is not in this match %', NEW.Player\_id,NEW.Match\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Tạo trigger

CREATE OR REPLACE TRIGGER check\_player\_eligibility\_trigger

BEFORE INSERT ON Match\_Player

FOR EACH ROW

EXECUTE FUNCTION check\_player\_eligibility();