**INDEX**

**Bảng Match**

1. Date\_of\_match: B-tree index. Useful for date-based queries, such as finding matches within a specific date range.

CREATE INDEX idx\_Date\_of\_match

USING btree

ON Match (Date\_of\_match);

SELECT \* FROM Match

WHERE Date\_of\_match BETWEEN date1 AND date2;

1. Referee\_main: B-tree index. Enhance query performance related to referee information.

(queries involving the most frequently selected main referees)

CREATE INDEX idx\_Referee\_main

USING btree

ON Match (Referee\_main);

SELECT Referee\_name, Referee\_main\_id, COUNT (\*) No\_match FROM Match

JOIN Referees ON Referee\_main\_id = Referee\_id

GROUP BY Referee\_main\_id

ORDER BY No\_match DESC

LIMIT 1;

**Bảng Player**

1. Nationality, Date\_of\_birth: B-tree index. If there are frequent queries based on nationality or date of birth.

(queries about which nationality has the most players currently playing, young players - U23)

CREATE INDEX idx\_Nationality\_Date\_of\_birth

USING btree

ON Player (Nationality, Date\_of\_birth);

SELECT Nationality, COUNT(\*) No\_players FROM Player

WHERE AGE(current\_date,Date\_of\_birth)<= Interval '23 year'

GROUP BY Nationality

ORDER BY No\_players DESC

LIMIT 1;

**Bảng Match\_Player**

1. Player\_id, Match\_id: Composite B-tree index. Useful for querying player information within each match.

(querying to find out how many assists and goals player A has in match B)

CREATE INDEX idx\_Player\_id\_Match\_id

USING btree

ON Match\_Player (Player\_id, Match\_id);

CREATE OR REPLACE FUNCTION total\_number\_of\_goals\_and\_assists(player\_name Varchar(30), team1 VARCHAR(30), team2 VARCHAR(30))

RETURNS INT AS $$

DECLARE total INT;

BEGIN

SELECT MP.goals+MP.assists FROM Match\_Player MP

INTO total

JOIN Player P USING(player\_id)

JOIN (Team\_join TJ JOIN Team T1 ON TJ.home\_team\_id=T1.team\_id

JOIN Team T2 ON TJ.away\_team\_id=T2.team\_id)

USING(match\_id)

WHERE P.name = player\_name AND T1.name=team1 AND T2.name=team2;

RETURN total;

END;

$$ LANGUAGE plpgsql;

**Bảng TEAM\_JOIN**

1. Home\_team\_id, Away\_team\_id: Composite B-tree index. Supporting queries for match results of specific teams.

CREATE INDEX idx\_Home\_team\_id\_Away\_team\_id

USING btree

ON TEAM\_JOIN (Home\_team\_id, Away\_team\_id);

CREATE OR REPLACE FUNCTION Show\_match(team1 VARCHAR(30), team2 VARCHAR(30))

RETURNS TABLE(

Match\_ids INT,

Home\_team VARCHAR(30),

Away\_team VARCHAR(30),

Score VARCHAR(30)

)

AS $$

BEGIN

RETURN QUERY

SELECT Match\_id,TA.Name, Tb.Name, CAST(CONCAT(home\_goals,' - ', away\_goals) AS VARCHAR) FROM Team\_join TJ

JOIN Team TA ON TA.Team\_id = TJ.Home\_team\_id

JOIN Team TB ON TB.Team\_id = TJ.Away\_team\_id

WHERE (TA.name = team1 AND TB.name = team2)

OR (TA.name = team2 AND TB.name = team1)

ORDER BY Match\_id;

END;$$

LANGUAGE plpgsql;

**Bảng Ticket**

1. Match\_id: B-tree index.

Supporting ticket queries by match.

(queries about which match has the highest revenue)

CREATE INDEX idx\_Match\_id

USING btree

ON Ticket (Match\_id);

CREATE OR REPLACE VIEW match\_has\_the\_highest\_revenue AS

SELECT Match\_id, Sum(number\_of\_sold\_tickets) Total\_number\_of\_sold\_tickets,

Sum(number\_of\_sold\_tickets\*Ticket\_price) as Total\_money

FROM Team\_join TJ

JOIN Ticket T USING(Match\_id)

Group By Match\_id

ORDER BY Total\_money DESC, Total\_number\_of\_sold\_tickets DESC;

**Reasons for Choosing B-tree Index:**

B-tree index is preferred due to its ability to support range-based queries, sorting, and efficient searching.

B-tree indexes perform well with both queries on a specific value and range-based queries.

Most cases in your database require the flexibility and performance that a B-tree index can provide, especially in queries involving date ranges, foreign keys, and data sorting.