**Football Team Management**

DBMS project

**Sabrina Lupsan**

Faculty of Cybernetics, Statistics and Economic Informatics

Coordinating professor: Diaconita Vlad

Series G

Group 1064

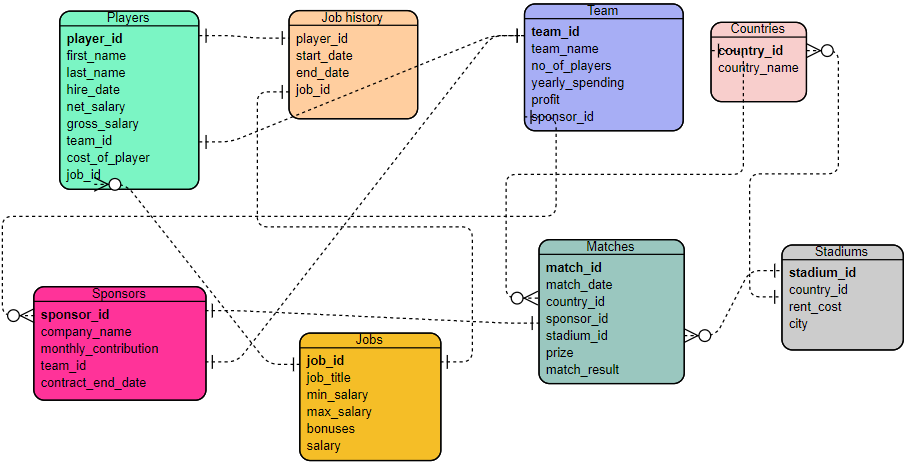
**Description of the theme**

I chose to make a database of a Football Team and learned how to manage it. The players have four different kind of jobs, based on real-life evaluation and strategies – goalkeeper, midfielder, defender and attacker. The minimum and maximum salary of each job depend on the job and the net salaries were calculated using the gross salary and the taxes applied. Each team has players, a sponsor, matches played. The matches were played in different stadiums, and those stadiums are in different countries. This is how it all ties together, bringing the management of a Football Team. I believe that such a database is useful for storing data for official competitions like Champions League or The World Championship, which need official data, and for media news, who need statistics, such as the ones displayed in the exercises given. There are large volumes of data in this field because there are hundreds of matches per week all around the world.

All of the teams and names were fictious and the salaries chosen for this database are not official or true.

**1. The conceptual schema of the database**

A database always begins with a conceptual schema, which helps to design and organize the coding part. This also helps identify **the unique keys** (in bold), foreign keys, connections to other tables in the database and, eventually, errors.



2. DDL and DML statements

--create table Players

BEGIN

EXECUTE IMMEDIATE

'CREATE TABLE Players

(

PLAYER\_ID NUMBER(6, 0) PRIMARY KEY,

LAST\_NAME VARCHAR2(50),

FIRST\_NAME VARCHAR2(50),

HIRE\_DATE DATE,

NET\_SALARY NUMBER(6),

GROSS\_SALARY NUMBER(6),

TEAM\_ID NUMBER(6, 0),

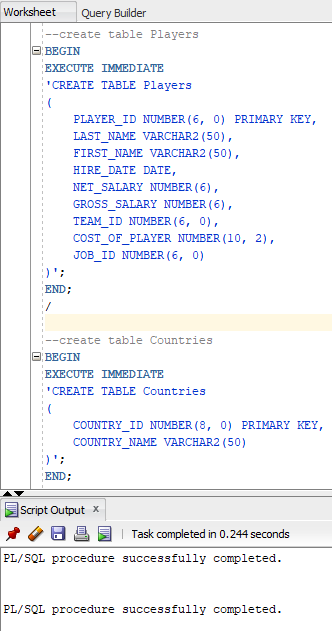
COST\_OF\_PLAYER NUMBER(10, 2),

JOB\_ID NUMBER(6, 0)

)';

END;

/



--create table Countries

BEGIN

EXECUTE IMMEDIATE

'CREATE TABLE Countries

(

COUNTRY\_ID NUMBER(8, 0) PRIMARY KEY,

COUNTRY\_NAME VARCHAR2(50)

)';

END;

/

--create table Stadiums

BEGIN

EXECUTE IMMEDIATE

'CREATE TABLE Stadiums

(

STADIUM\_ID NUMBER(8, 0) PRIMARY KEY,

COUNTRY\_ID NUMBER(8, 0),

RENT\_COST NUMBER(5, 2),

CITY VARCHAR2(20)

)';

END;

/

--create table Team

BEGIN

EXECUTE IMMEDIATE

'CREATE TABLE Team

(

TEAM\_ID NUMBER(6, 0) PRIMARY KEY,

TEAM\_NAME VARCHAR2(100),

NUMBER\_OF\_PLAYERS NUMBER(3),

YEARLY\_SPENDINGS NUMBER(8, 2),

PROFIT NUMBER(5, 2),

SPONSOR\_ID NUMBER(5, 2)

)';

END;

/

--create table Jobs

BEGIN

EXECUTE IMMEDIATE

'CREATE TABLE Jobs

(

JOB\_ID NUMBER(8, 0) PRIMARY KEY,

JOB\_TITLE VARCHAR2(50),

MIN\_SALARY NUMBER(8, 2),

MAX\_SALARY NUMBER(8, 2),

BONUSES NUMBER(6, 2),

SALARY NUMBER(8, 2)

)';

END;

/

--create table Job\_History

BEGIN

EXECUTE IMMEDIATE

'CREATE TABLE Job\_History

(

JOB\_ID NUMBER(8, 0),

START\_DATE DATE,

END\_dATE DATE,

PLAYER\_ID NUMBER(8, 0)

)';

END;

/

--create table Sponsors

BEGIN

EXECUTE IMMEDIATE

'CREATE TABLE Sponsors

(

SPONSOR\_ID NUMBER(8, 0) PRIMARY KEY,

COMPANY\_NAME VARCHAR2(200),

MONTHLY\_CONTRIBUTION NUMBER(6,2),

TEAM\_ID NUMBER(6),

CONTRACT\_END\_DATE DATE

)';

END;

/

--create table Match

BEGIN

EXECUTE IMMEDIATE

'CREATE TABLE Match

(

MATCH\_ID NUMBER(8, 0) PRIMARY KEY,

MATCH\_DATE DATE,

COUNTRY\_ID NUMBER(8,0),

SPONSOR\_ID NUMBER(8,0),

PRIZE NUMBER(5, 0),

MATCH\_RESULT VARCHAR(50)

)';

END;

/

--rename the table "Match" to "Matches"

BEGIN

EXECUTE IMMEDIATE

'ALTER TABLE Match RENAME TO Matches';

END;

/

--drop the column "FIRST\_NAME" from the Players table

BEGIN

EXECUTE IMMEDIATE

'ALTER TABLE Players DROP COLUMN FIRST\_NAME';

END;

/

--update the net salary of each player by deducting social (25%) and health (10%) insurance

--and the income tax (10%) from the gross salary

BEGIN

UPDATE PLAYERS

SET NET\_SALARY = 0.9\*(GROSS\_SALARY - 0.35\*GROSS\_SALARY);

END;

/

--divide the players into 2 teams: players with the player\_id in the interval [1, 2, 3, 4] are in the

--team with the id=5555 and players with IDs 5 and 6 are in the team with the team\_id = 6666

BEGIN

UPDATE PLAYERS

SET TEAM\_ID = 5555 WHERE PLAYER\_ID IN (1, 2, 3, 4);

UPDATE PLAYERS

SET TEAM\_ID = 6666 WHERE PLAYER\_ID IN (5, 6);

END;

/

--Display in how much time the contract of every sponsor will expire, in intervals of: <1 year, 1-2 years, > 2 years.

SET SERVEROUTPUT ON

DECLARE

CURSOR end\_date IS SELECT company\_name,

(CASE

WHEN EXTRACT(YEAR FROM contract\_end\_date) - EXTRACT(YEAR FROM SYSDATE) < 1 THEN 'Less than 1 year'

WHEN EXTRACT(YEAR FROM contract\_end\_date) - EXTRACT(YEAR FROM SYSDATE) BETWEEN 1 AND 2 THEN 'Between 1 and 2 years'

ELSE 'More than 2 years'

END) YEARS

FROM Sponsors

ORDER BY (EXTRACT(YEAR FROM contract\_end\_date));

company end\_date%rowtype;

BEGIN

OPEN end\_date;

LOOP

FETCH END\_DATE INTO company;

EXIT WHEN END\_DATE%NOTFOUND;

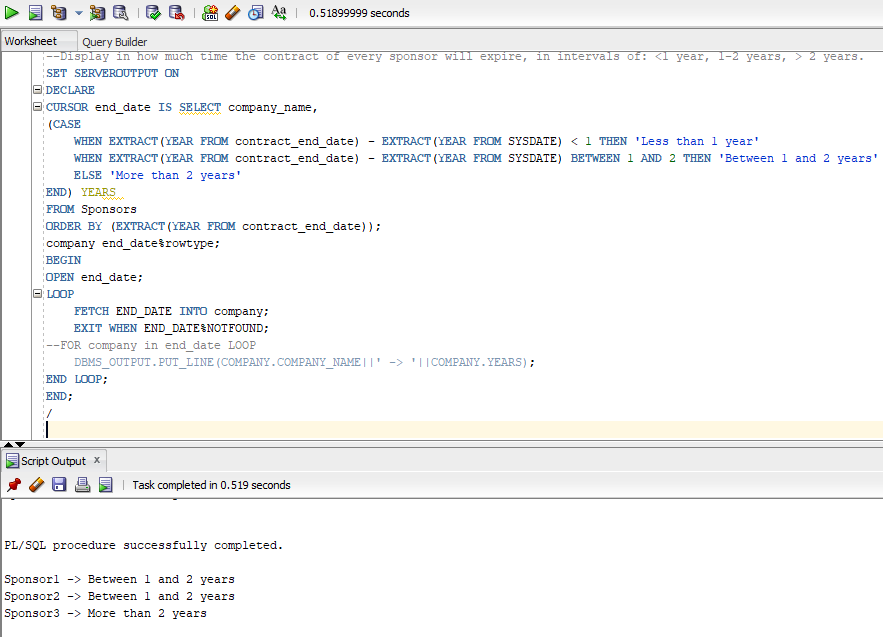
DBMS\_OUTPUT.PUT\_LINE(COMPANY.COMPANY\_NAME||' -> '||COMPANY.YEARS);

END LOOP;

CLOSE end\_date;

END;

/



--compute the value of the net salaries that a team would have to pay if it would only have players with even IDs

SET SERVEROUTPUT ON

DECLARE

type player is record (

player\_id number(8, 2),

net\_salary number(10, 2)

);

v\_player player;

v\_total number(10, 0):=0;

v\_players number(4, 0):=0;

BEGIN

SELECT COUNT(PLAYER\_ID) INTO v\_players FROM PLAYERS;

for i in 1..v\_players loop

if i mod 2 = 0 then

select player\_id, net\_salary into v\_player from players where player\_id = i;

v\_total := v\_total + v\_player.net\_salary;

end if;

end loop;

dbms\_output.put\_line('The net salary of players with even player\_id is: '||v\_total);

end;

/

--display the name and player ID of the player in the 'Rangers' team. If there is more than one, print that

SET SERVEROUTPUT ON

DECLARE

v\_name VARCHAR2(50);

v\_id NUMBER(8);

BEGIN

SELECT last\_name, player\_id INTO v\_name, v\_id FROM PLAYERS WHERE TEAM\_ID = (SELECT TEAM\_ID FROM TEAM WHERE team\_name = 'Rangers');

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('There is more than one player');

END;

/

SET SERVEROUTPUT ON

DECLARE

CURSOR CITIES IS SELECT STADIUM\_ID, CITY FROM STADIUMS;

acity CITIES%ROWTYPE;

BEGIN

FOR acity in CITIES LOOP

OPEN CITIES;

DBMS\_OUTPUT.PUT\_LINE('Stadium '||acity.stadium\_id||' is in: '||acity.CITY);

END LOOP;

EXCEPTION WHEN CURSOR\_ALREADY\_OPEN THEN

DBMS\_OUTPUT.PUT\_LINE('There was an error handling the cursor.');

END;

/

--Display the matches with the prize bigger than a given value dollars but exclude the ones in which the guest team did not win or they tied.

SET SERVEROUTPUT ON

DECLARE

CURSOR PRIZES(p\_prize number) IS SELECT MATCH\_ID, PRIZE, MATCH\_DATE, MATCH\_RESULT FROM MATCHES WHERE prize > p\_prize;

v\_prize prizes%rowtype;

BEGIN

FOR v\_prize in prizes(10000) LOOP

if v\_prize.match\_result <> 'Host team lost' THEN

DBMS\_OUTPUT.PUT\_LINE(v\_prize.match\_id||' '||v\_prize.prize||' '||v\_prize.match\_date||' '||v\_prize.match\_result);

END IF;

END LOOP;

END;

/

--display all jobs that have bonuses lower than 2000

SET SERVEROUTPUT ON

DECLARE

no\_field exception;

pragma exception\_init(no\_field, -20101);

BEGIN

UPDATE Sponsors

SET sponsor\_id = 0

WHERE company\_name LIKE '%Oracle';

IF SQL%NOTFOUND = TRUE THEN

RAISE\_APPLICATION\_ERROR(-20101, 'No such sponsor found.');

END IF;

EXCEPTION

WHEN no\_field THEN

DBMS\_OUTPUT.PUT\_LINE('No such sponsor found');

END;

/

--create a package

create or replace package player\_pack\_1 is

--display the number of players in a certain team

function get\_no\_players(v\_team\_id number) return number;

--Estimate the average amount of money a future player can win per month given the job\_id

function average\_salary(v\_job\_id number) return number;

--show the date of the match with a given id

function date\_of\_match(v\_match\_id number) return date;

--increase the salary of a certain player

procedure increase\_salary(v\_player\_id number, v\_percentage number);

--change the contract of ‘Sponsor2’ to end on the current date

procedure contract\_ends(v\_name string);

end;

create or replace package body player\_pack\_1 is

function get\_no\_players(v\_team\_id number) return number is

v\_number number;

begin

select count(\*) into v\_number from players where team\_id = v\_team\_id;

return v\_number;

end;

function average\_salary(v\_job\_id number) return number is

v\_avg\_salary number:=0;

begin

select round(((max\_salary - min\_salary)/2+bonuses)/12) into v\_avg\_salary

from jobs where job\_id = v\_job\_id;

return v\_avg\_salary;

end;

function date\_of\_match(v\_match\_id number) return date is

v\_date date;

begin

select match\_date into v\_date from matches where v\_match\_id = match\_id;

return v\_date;

end;

procedure increase\_salary(v\_player\_id number, v\_percentage number) is

no\_player exception;

pragma exception\_init(no\_player, -20101);

begin

update players

set net\_salary = net\_salary + net\_salary\*v\_percentage/100

where player\_id = v\_player\_id;

if sql%notfound = true then

raise\_application\_error(-2101, 'There was no player found');

end if;

exception when no\_player then

null;

end;

procedure contract\_ends(v\_name string) is

v\_end date;

begin

select sysdate into v\_end from dual;

update sponsors

set contract\_end\_date = v\_end

where company\_name = v\_name;

end;

end;

set SERVEROUTPUT on;

begin

dbms\_output.put\_line('The number of players from the requested team is '||player\_pack.get\_no\_players(5555));

dbms\_output.put\_line('The average monthly salary for the requested job is '||player\_pack.average\_salary(400));

dbms\_output.put\_line('The date of the requested match is '||player\_pack.date\_of\_match(12));

player\_pack.increase\_salary(78, 5);

player\_pack.increase\_salary(3, 5);

player\_pack.contract\_ends('Sponsor3');

EXCEPTION

WHEN OTHERS THEN

dbms\_output.put\_line('Wrong data.');

end;

/

--create a trigger that doesn't allow a player to be in a different team than 5555 and 6666

CREATE OR REPLACE TRIGGER non\_existent\_team BEFORE INSERT ON PLAYERS for each row

BEGIN

IF :new.team\_id not in (5555, 6666) then

raise\_application\_error(-20001, 'Wrong team ID');

end if;

END;

/

--to check if the trigger is triggered

INSERT INTO Players (PLAYER\_ID, LAST\_NAME, HIRE\_DATE, NET\_SALARY, GROSS\_SALARY, TEAM\_ID, COST\_OF\_PLAYER, JOB\_ID) VALUES (90, 'Andrei', '19-NOV-2019', 9000, 9100, 100, 100000, 400);

--construct a trigger that won't allow you to input a hire\_date that is in the future

CREATE OR REPLACE TRIGGER invalid\_time BEFORE INSERT OR UPDATE ON PLAYERS for each row

BEGIN

IF :new.hire\_date > sysdate then

raise\_application\_error(-20002, 'The date is not valid.');

end if;

END;

/

--to check

INSERT INTO Players (PLAYER\_ID, LAST\_NAME, HIRE\_DATE, NET\_SALARY, GROSS\_SALARY, TEAM\_ID, COST\_OF\_PLAYER, JOB\_ID) VALUES (90, 'Andrei', '19-NOV-2021', 9000, 9100, 5555, 100000, 400);

**APEX part**

Link: <https://rpgzn0h8icu4sy6-sgbd.adb.eu-frankfurt-1.oraclecloudapps.com/ords/f?p=106:1:113874474989329:::::>

