

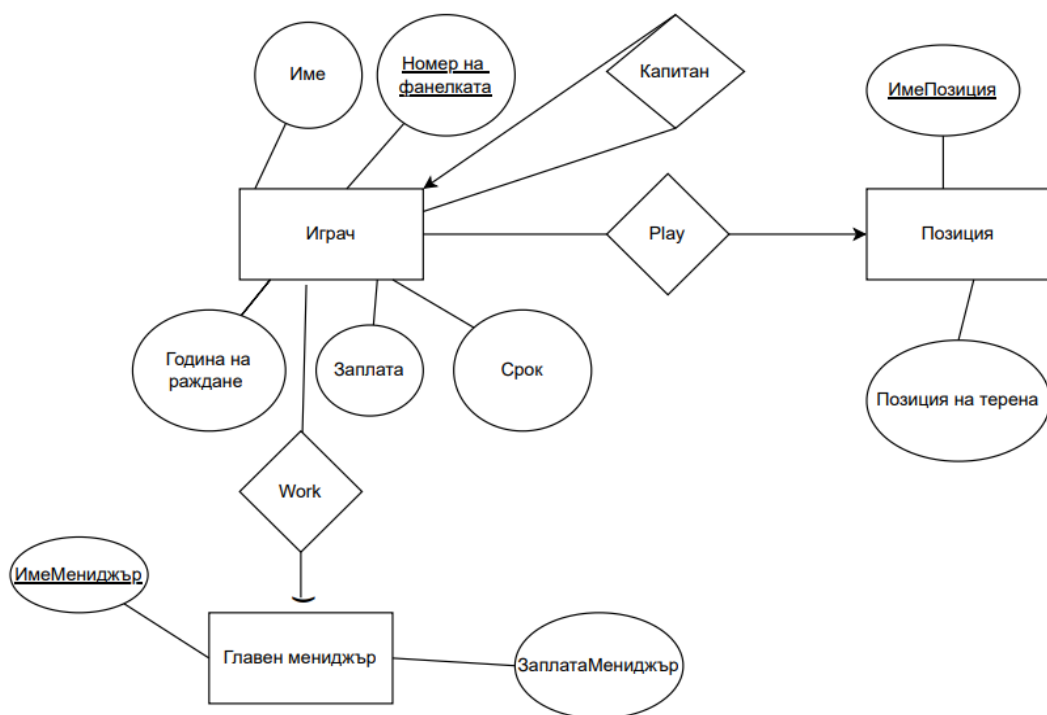
Тема „ База от данни за играчите в един футболен отбор”

Изготвил: Данаил Иванов ф.н. 45784

1.Обхват на модела. Дефинирайте задачата.

Информационна система съхранява информация за играчите в един футболен отбор. За един един играч(футболист) от футболен отбор се пазят следните данни: име, номер на фанелката(уникален идентификатор),година на раждане, заплата, срок на договора. Всеки играч си има длъжностна позиция в отбора. Като на една позиция може да играят много играчи, а един играч играе само на една позиция(в редки случай на контузия може и на различна, но няма да разглеждаме този случай). За всяка позиция се пази информация за името й(уникално) и позицията на терена. В един футболен отбор всеки играч може да бъде капитан на отбора, но капитана може да бъде само един. Всеки играч тренира и играе под наставленията на един главен мениджър, който ръководи всички играчи. Той се характеризира с уникално име и заплата.

2. E/R модел на данни



3. Релационен модел на данни

Игратч(име, номер на фанелка, година на разждане, заплата, срок,
ИмеМениджър, ИмеПозиция, НомерКапитан)
 Позиция(ИмеПозиция, Позиция на терена)
 ГлавенМениджър(ИмеМениджър, ЗаплатаМениджър)

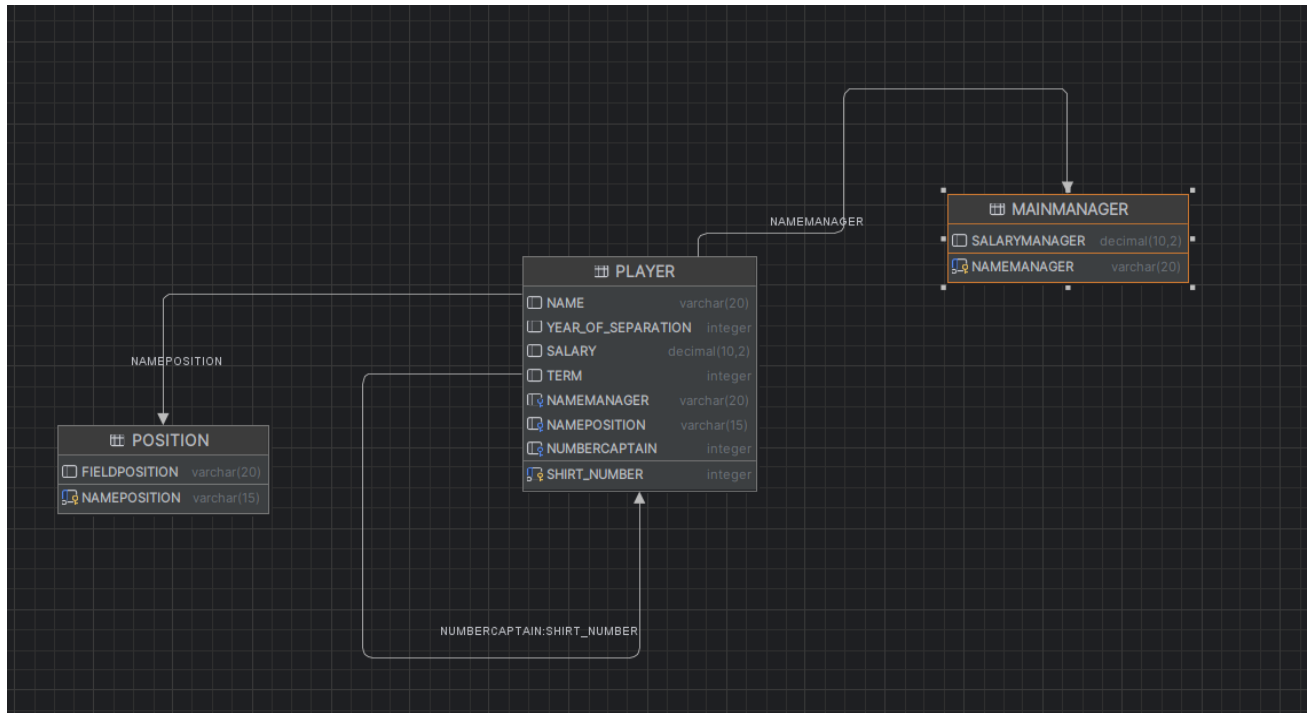
Ограничения

Игратч(номер на фанелка)
 Позиция(ИмеПозиция)
 ГлавенМениджър(ИмеМениджър)

Foreign keys

Игратч: FK(ИмеМениджър) -> ГлавенМениджър (ИмеМениджър)
 Игратч: FK(ИмеПозиция) -> Позиция (ИмеПозиция)
 Игратч: FK(НомерКапитан) -> Игратч(НомерКапитан)

4. Схема на базата от данни



5. Функции, тригери и изгледи

Функции

Описание на функция 1:

Функцията `f_get_maneger` е SQL функция, която извлича името на мениджъра за даден номер на фланелка на играч

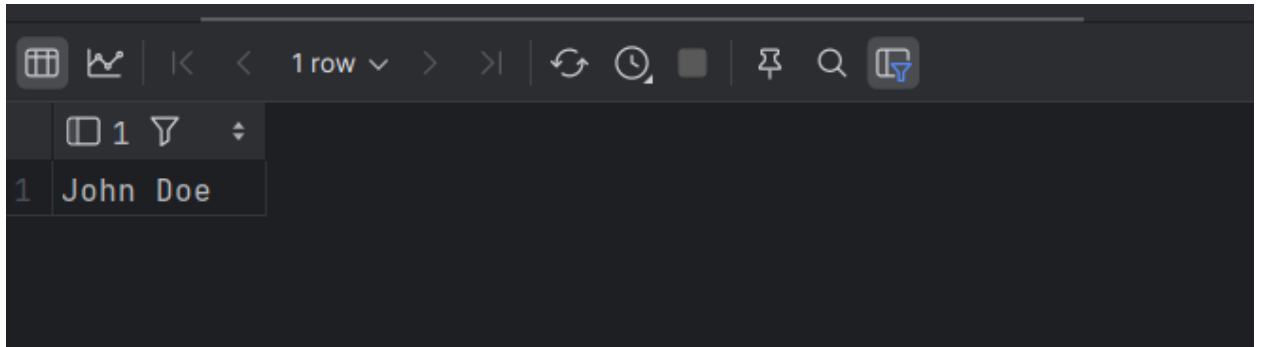
SQL:

```
create or replace function f_get_maneger (p_shirt int)
returns varchar(20)
begin
    declare name varchar(20);
    set name = (select NAMEMANAGER from player where SHIRT_NUMBER = p_shirt);
    return name;
end;
```

Извикване на функцията:

```
values FN24 45784.f_get_maneger(1);
```

Резултат:



The screenshot shows a database interface with a toolbar at the top containing icons for table view, chart, navigation, and search. Below the toolbar, a table displays the result of a query. The table has one row with the name 'John Doe'.

1	John Doe
---	----------

Описание на функция 2: Функцията `f_get_player_on_position` е SQL функция, която връща информация за играчите на определена позиция.

SQL:

```
create or replace function f_get_player_on_position (p_position varchar(15))  
returns table (name varchar(20), shirt_number int, salary decimal(20,2))  
return (select name, shirt_number, salary  
        from PLAYER  
        where NAMEPOSITION = p_position);
```

Извикване на функцията:

```
select * from table(FN24 45784.F_GET_PLAYER_ON_POSITION('Forward')) t;
```

Резултат:

	NAME	SHIRT_NUMBER	SALARY
1	Player One	1	30000.00
2	Player Five	5	30000.00
3	Player Nine	9	30000.00
4	Player Thirteen	13	32000.00
5	Player Seventeen	17	36000.00

Тригери

Описание на тригер 1: Това е SQL тригер, който гарантира, че заплатата на играч е минимум 30000.00 лв.

SQL:

```
CREATE OR REPLACE TRIGGER BeforeInsertEnsureMinimumSalary
BEFORE INSERT ON Player
REFERENCING NEW AS NEWROW
FOR EACH ROW
BEGIN
    IF NEWROW.salary < 30000.00 THEN
        SET NEWROW.salary = 30000.00; -- Minimum salary
    END IF;
END;
```

Тестване на тригер 1:

```
INSERT INTO Player (name, shirt_number, year_of_separation, salary, term,
NameManager, NamePosition, NumberCaptain)
VALUES ('Goshko', 69, 1990, 12000, null, 'John Doe', 'Forward', 1);
```

Описание на тригер 2:

Този тригер е SQL код, който се изпълнява преди всяко вмъкване на запис в таблицата MAINMANAGER. Целта му е да ограничи максималната заплата на мениджърите до 1 000 000.00 лв.

SQL:

```
create or replace trigger tr_before_insert_sal_menager
before insert on MAINMANAGER
referencing new as n
for each row
begin
    if n.SALARYMANAGER > 1000000 then
        set n.SALARYMANAGER = 1000000;
    end if;
end;
```

Тестване на тригер 2:

```
insert into MAINMANAGER (namemanager, salarymanager) values ('Ivancho',
1000002);
```

Изгледи:

Описание на изглед 1:

Този SQL код дефинира изглед, наречен PlayerPositions, който показва информация за играчите и техните позиции.

SQL:

```
CREATE or replace VIEW PlayerPositions AS
SELECT name AS player_name, shirt_number, NamePosition
FROM Player;
```

Извикване на изглед 1:

```
SELECT * FROM PlayerPositions;
```

Резултат:

	PLAYER_NAME ▾	SHIRT_NUMBER ▾	NAMEPOSITION ▾
1	Player One		1 Forward
2	Player Two		2 Defender
3	Player Three		3 Midfielder
4	Player Four		4 Goalkeeper
5	Player Five		5 Forward

Описание на изглед 2:

Този SQL код дефинира изглед, наречен ManagerPlayerCount, който показва броя на играчите за всеки мениджър.





SQL:

```
CREATE OR REPLACE VIEW ManagerPlayerCount AS
SELECT NameManager, COUNT(*) AS player_count
FROM Player
GROUP BY NameManager;
```

Извикване на изглед 2:

```
SELECT * FROM ManagerPlayerCount;
```

Резултат:

	 NAMEMANAGER 	 PLAYER_COUNT 
1	Alice Johnson	5
2	Bob Brown	4
3	Jane Smith	5
4	John Doe	7

6. Приложение за достъп до базата

Java код:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class DB2Test {

    private Connection connection;
    private Statement statement;
    private ResultSet resultSet;

    public void openConnection(){

// Step 1: Load IBM DB2 JDBC driver

        try {

            DriverManager.registerDriver(new com.ibm.db2.jcc.DB2Driver());

        }

        catch(SQLException cnfex) {
```

```

        System.out.println("Problem in loading or registering IBM DB2
JDBC driver");

        cnfex.printStackTrace();
    }

// Step 2: Opening database connection

    try {

        connection =
DriverManager.getConnection("jdbc:db2://62.44.108.24:50000/SAMPLE",
"db2admin", "db2admin");

        statement = connection.createStatement();

    }

    catch(SQLException s){

        s.printStackTrace();

    }

}

public void closeConnection(){

    try {

        if(null != connection) {

            // cleanup resources, once after processing

            resultSet.close();

            statement.close();

            // and then finally close connection

            connection.close();

        }

    }

    catch (SQLException s) {

        s.printStackTrace();

    }

}

public void select(String stmtnt, int column) {

```



```

        try{
            resultSet = statement.executeQuery(stmtnt);

            String result = "";

            while(resultSet.next()) {

                for (int i = 1; i <= column; i++) {

                    result += resultSet.getString(i);

                    if (i == column) result += " \n";
                    else result += ", ";

                }

                System.out.println("Executing query: " + stmtnt + "\n");
                System.out.println("Result output \n");
                System.out.println("----- \n");
                System.out.println(result);

            }
        } catch (SQLException s)
        {
            s.printStackTrace();
        }
    }

    public void insert(String stmtnt) {

        try{

            statement.executeUpdate(stmtnt);

        }

        catch (SQLException s){

            s.printStackTrace();

        }

        System.out.println("Successfully inserted!");

    }

    public void delete(String stmtnt) {

        try{

            statement.executeUpdate(stmtnt);

        }
    }

```

```

        catch (SQLException s){

            s.printStackTrace();

        }

        System.out.println("Successfully deleted!");
    }

    public static void main(String[] args) {

        DB2Test db2Obj = new DB2Test();
        String stmtnt = "";
        db2Obj.openConnection();

        stmtnt = "SELECT NAME, SHIRT_NUMBER, SALARY FROM FN24_45784.PLAYER";

        db2Obj.select(stmtnt, 3);

        db2Obj.closeConnection();

    }}

```

Резултат:

Executing query: SELECT NAME, SHIRT_NUMBER, SALARY FROM FN24_45784.PLAYER

Result output

Player One, 1, 20000.00
 Player Two, 2, 22000.00
 Player Three, 3, 21000.00
 Player Four, 4, 23000.00
 Player Five, 5, 24000.00
 Player Six, 6, 25000.00
 Player Seven, 7, 26000.00
 Player Eight, 8, 27000.00
 Player Nine, 9, 28000.00
 Player Ten, 10, 29000.00
 Player Eleven, 11, 30000.00
 Player Twelve, 12, 31000.00
 Player Thirteen, 13, 32000.00
 Player Fourteen, 14, 33000.00

Player Fifteen, 15, 34000.00
Player Sixteen, 16, 35000.00
Player Seventeen, 17, 36000.00
Player Eighteen, 18, 37000.00
Player Nineteen, 19, 38000.00
Player Twenty, 20, 39000.00