**ASSIGNMENT 2 FRONT SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **TEC Level 5 HND Diploma in Computing** | | |
| **Unit number and title** | **Unit 04: Database Design & Development** | | |
| **Submission date** |  | **Date Received 1st submission** |  |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** | Nguyen Trong Duy | **Student ID** | GCD17313 |
| **Class** |  | **Assessor name** |  |
| **Student declaration**  I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice. | | | |
|  |  | **Student’s signature** | Duy |

**Grading grid**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P2 | P3 | P4 | P5 | M2 | M3 | M4 | M5 | D2 | D3 |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **❒ Summative Feedback: ❒ Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Signature & Date:** | | |

# table of contents

Contents

[table of contents 1](#_Toc108270092)

[I. DEVELOP THE DATABASE SYSTEM (P2) 2](#_Toc108270093)

[1. FINAL MOCK-UP OF THE APPLICATION 2](#_Toc108270094)

[2. OUTPUTS AND DATA VALIDATION 8](#_Toc108270095)

[2.1 QUERIES TO CREATE DATABASE AND TABLE 8](#_Toc108270096)

[2.2 DATA VALIDATION 11](#_Toc108270097)

[II. PRODUCE QUERIES 17](#_Toc108270098)

[1. QUERIES TO INSERT DATA WITH ILLUSTRATIONS OF FINAL RESULT 17](#_Toc108270099)

[2. QUERIES TO SELECT DATA WITH ILLUSTRATIONS OF FINAL RESULT 19](#_Toc108270100)

[3. QUERIES TO UPDATE DATA WITH ILLUSTRATIONS OF FINAL RESULT 21](#_Toc108270101)

[4. QUERIES TO DELETE DATA WITH ILLUSTRATIONS OF FINAL RESULT 22](#_Toc108270102)

[5. ADVANCED QUERIES: STORED PROCEDURES, TRIGGERS, FUNCTIONS 23](#_Toc108270103)

[III. TEST THE SYSTEM 29](#_Toc108270104)

[1. TEST PLAN 29](#_Toc108270105)

[2. TEST RESULT 31](#_Toc108270106)

[IV. PRODUCE TECHNICAL AND USER DOCUMENTATION (P5) 33](#_Toc108270107)

[1. TECHNICAL DOCUMENT 33](#_Toc108270108)

[1.1 INTRODUCTION 33](#_Toc108270109)

[1.2 USER REQUIREMENT 34](#_Toc108270110)

[1.3 DIAGRAM 35](#_Toc108270111)

[1.4 POSSIBLE AND REALISTIC IMPROVEMENT 36](#_Toc108270112)

[1.5 CURRENT VERSION NUMBER 36](#_Toc108270113)

[1.6 DATE LAST CHANGED AND IMPLEMENTED 36](#_Toc108270114)

[2. USER DOCUMENT 37](#_Toc108270115)

[Menu window 37](#_Toc108270116)

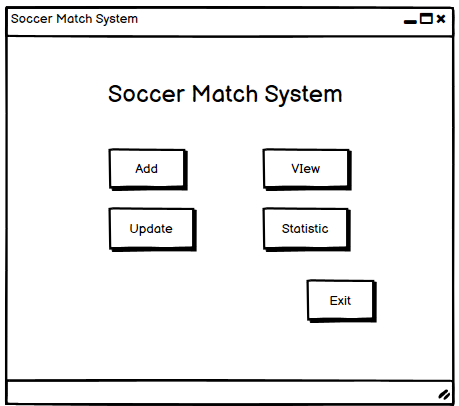
[REFERENCE LIST 45](#_Toc108270117)

# I. DEVELOP THE DATABASE SYSTEM (P2)

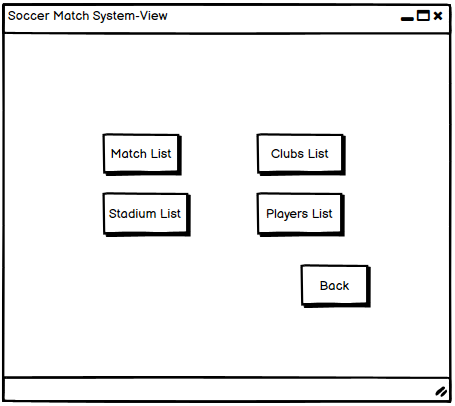
## 1. FINAL MOCK-UP OF THE APPLICATION

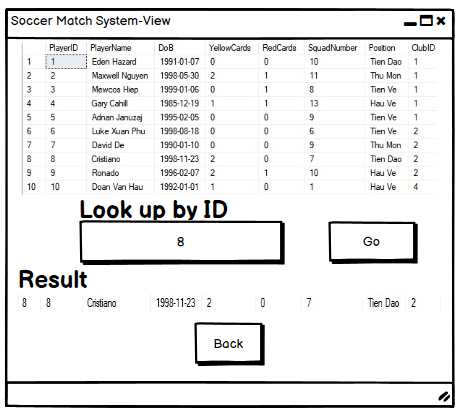
❖ In the first report, I went into detail about the restaurant management system's users and features. In this report, I have designed a user interface that integrates the system's features in order to develop and perfect the system. And here are the football management system's interfaces:

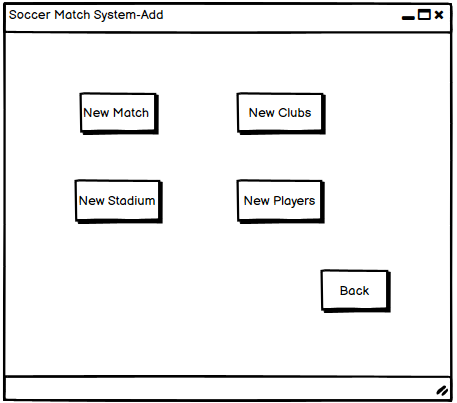
• The first is the system's main interface, which includes View, Add, Update, Statistic, and Exit buttons.



• Interface when the user clicks the View button. Items related to information of match, clubs, stadium, players, etc. will be displayed, users can click on the option to view information.

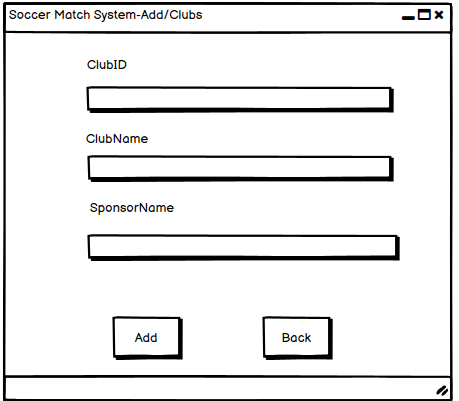


• When the user selects View Clubs List, the interface will appear as follows. 

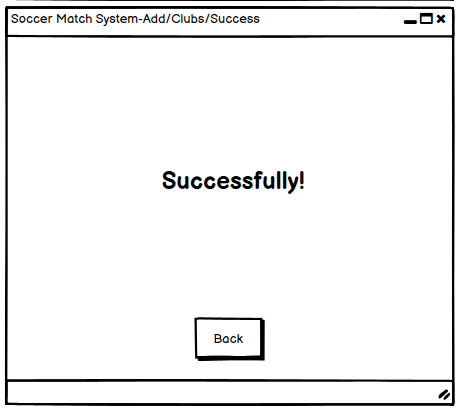


• The interface when the user clicks the Add button. Items related to more information of match, clubs, stadium, players, etc. will be displayed, users can click on options to add necessary information.

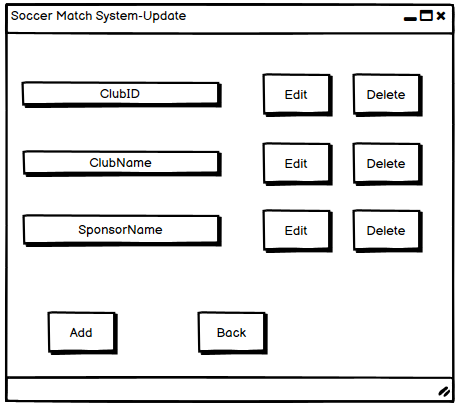
• When the user selects Add Clubs, the required information will be displayed. After the information is filled in, the user will press the Add button to complete the job.



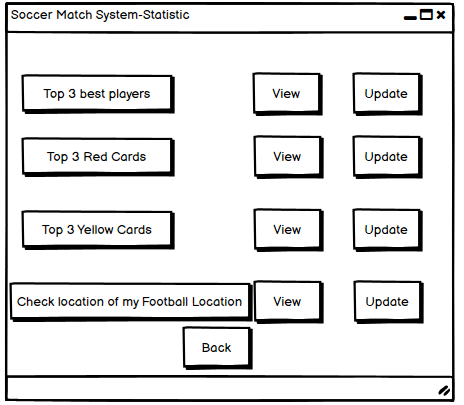
* When the Add is successful, the system will display the message "Successfully!!!".



• The interface when the user clicks the Update button. Items related to additional information of ID, name, SponsorName etc. will be displayed, users can click on options to edit necessary information.



• The interface when the user clicks the Statistic button. The statistics and other functions of the system will be displayed. Users can choose View or Update for new updates.



## 2. OUTPUTS AND DATA VALIDATION

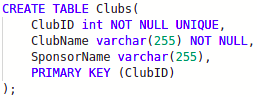
### 2.1 QUERIES TO CREATE DATABASE AND TABLE

- When starting to create a database for the system, the first thing to do is to create a database to store information about tables.

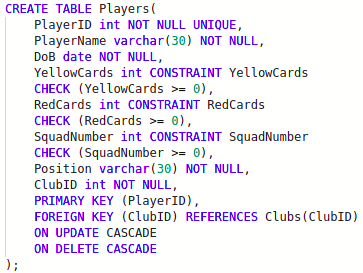


*Figure 1: Create database for system*

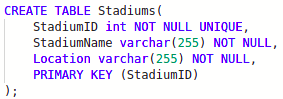
- Next, we create the tables according to the original design



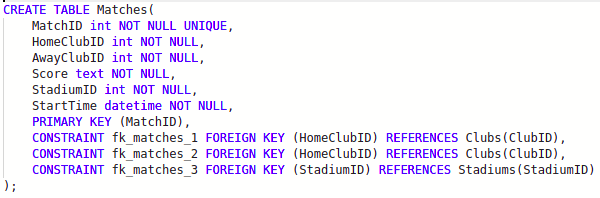
*Figure 1: Create Clubs table*



*Figure 1: Create Players table*



*Figure 1: Create Stadiums table*

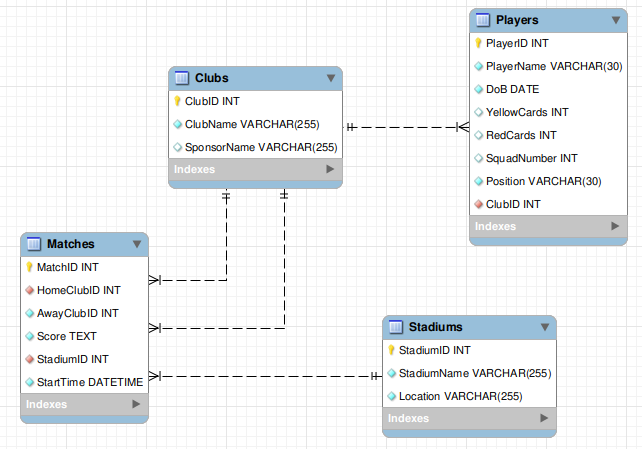


*Figure 1: Create Matches table*

- Primary and foreign keys will be generated while creating table. After create tables, we will get the result as shown below:



*Figure 1: Result of table creation*

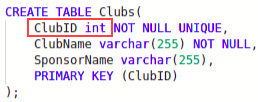


*Figure 1: Database diagram of system*

### 2.2 DATA VALIDATION

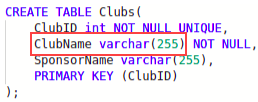
**\* Data type**

- int: A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The size parameter specifies the maximum display width (which is 255)



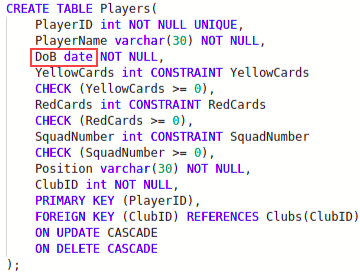
*Figure 1: Example of “int”**data type*

- varchar: A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535



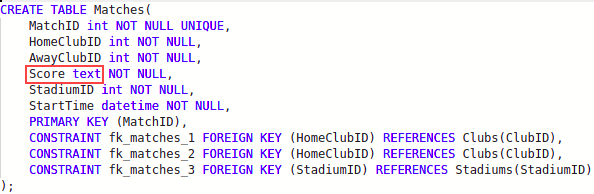
*Figure 1: Example of “*varchar*”**data type*

- date: A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'



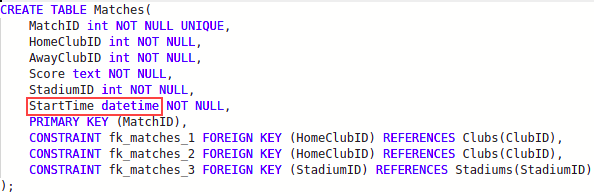
*Figure 1: Example of “*date*”**data type*

- text: Holds a string with a maximum length of 65,535 bytes

**

*Figure 1: Example of “*text*”**data type*

- datetime: A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time

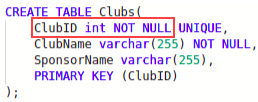
**

*Figure 1: Example of “*timestamp*”**data type*

**\* Presence**

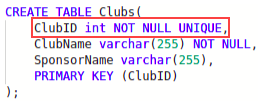
CONSTRAINT is used to define rules to allow or restrict what values can be stored in columns. The purpose of inducing constraints is to enforce the integrity of a database

- NOT NULL: NOT NULL constraint allows to specify that a column can not contain any NULL value. NOT NULL can be used to CREATE and ALTER a table



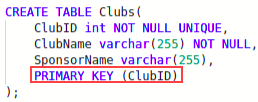
*Figure 1: Example of “NOT NULL”**constraint*

- UNIQUE: The UNIQUE constraint does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table

**

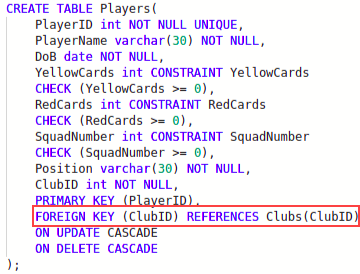
*Figure 1: Example of “UNIQUE”**constraint*

- PRIMARY KEY: A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint creates a unique index for accessing the table faster



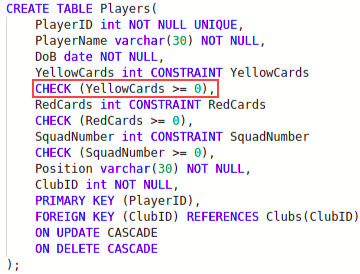
*Figure 1: Example of “PRIMARY KEY”**constraint*

- FOREIGN KEY: A FOREIGN KEY creates a link between two tables by one specific column of both tables. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY

**

*Figure 1: Example of “FOREIGN KEY”**constraint*

- CHECK: A CHECK constraint controls the values in the associated column. The CHECK constraint determines whether the value is valid or not from a logical expression

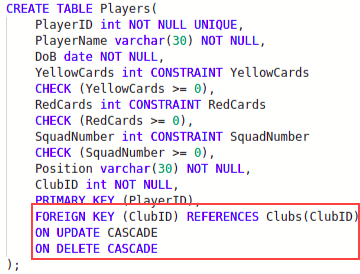
**

*Figure 1: Example of “CHECK”**constraint*

**\* Data integrity**

- CASCADE: Delete or update the row from the parent table and automatically delete or update the matching rows in the child table. Both ON DELETE CASCADE and ON UPDATE CASCADE are supported. Between two tables, do not define several ON UPDATE CASCADE clauses that act on the same column in the parent table or in the child table.

If a FOREIGN KEY clause is defined on both tables in a foreign key relationship, making both tables a parent and child, an ON UPDATE CASCADE or ON DELETE CASCADE subclause defined for one FOREIGN KEY clause must be defined for the other in order for cascading operations to succeed. If an ON UPDATE CASCADE or ON DELETE CASCADE subclause is only defined for one FOREIGN KEY clause, cascading operations fail with an error.

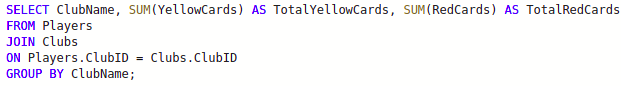


*Figure 1: Example of “CASCADE” data integrity*

With the above table creation sql statement, we have defined ON UPDATE CASCADE and ON DELETE CASCADE for the foreign key of the Players table. When we update the ClubID of the Clubs table, the data of the Players table (ie ClubID) will also change. If we delete information containing ClubID in the Clubs table, the data of the Players table will delete all records related to ClubID that have been deleted.- Querying Across Multiple Tables: “JOIN” statement is used to querying across multiple tables. A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

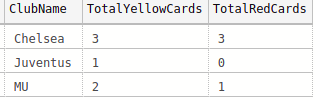
Here are the different types of the JOINs in SQL:

* (INNER) JOIN: Returns records that have matching values in both tables
* LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



*Figure 1: Example of “Querying Across Multiple Tables” data integrity*

This query is used to count the number of yellow and red cards of all players in a club. We need to use the “JOIN” statement to link the Players and Clubs tables together based on the ClubID. Next, group them together by ClubName via the “GROUP BY” statement. After that, need to use “SELECT” statement to get ClubName and calculate total yellow and red cards.

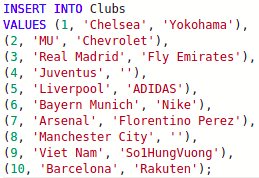


*Figure 1: Result of “Querying Across Multiple Tables”*

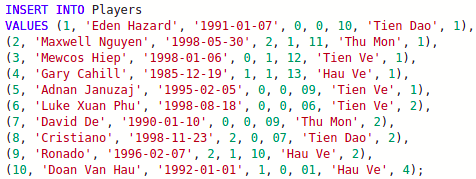
# II. PRODUCE QUERIES

## 1. QUERIES TO INSERT DATA WITH ILLUSTRATIONS OF FINAL RESULT

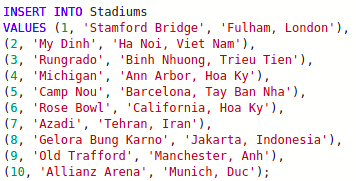
- After creating the table successfully, we need to insert data information into the tables



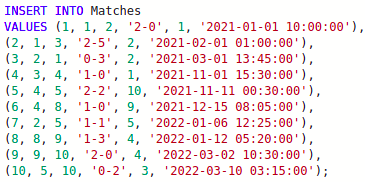
*Figure 1: Insert data into “Clubs” table*

**

*Figure 1: Insert data into “Players” table*

**

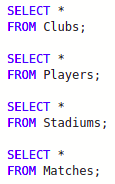
*Figure 1: Insert data into “Stadiums” table*

**

*Figure 1: Insert data into “Matches” table*

## 2. QUERIES TO SELECT DATA WITH ILLUSTRATIONS OF FINAL RESULT

- After performing insert data into the database, we can check the information that has been inserted into the database by using the “SELECT” statement.

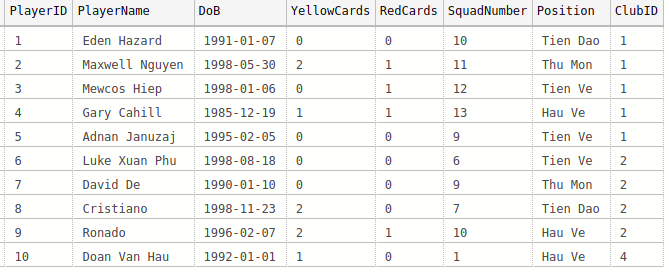


*Figure 1: Select data of all tables*

- Data information of all tables will be displayed as follows:



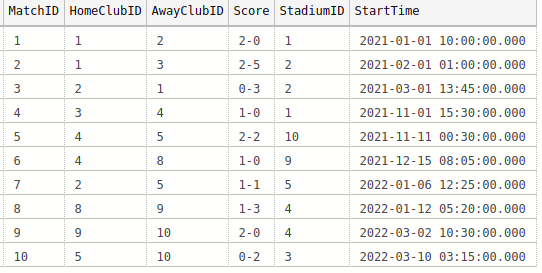
*Figure 1: Data information of Clubs table*

**

*Figure 1: Data information of Players table*

**

*Figure 1: Data information of Stadiums table*

**

*Figure 1: Data information of Matches table*

## 3. QUERIES TO UPDATE DATA WITH ILLUSTRATIONS OF FINAL RESULT

- During the system management process, if the information inserted into the DB is incorrect, need to correct the information or want to update the information after it has been inserted, we use the "UPDATE" statement.

Example:

+ User wants to update some information of a player with id = 3 such as wearing shirt number 8, born on "1999-01-06", we write the following query:

*Figure 1: Example of query update data*

+ Use “SELECT” statement to show data before update

*Figure 1: Data before update*

+ Use “SELECT” statement to show data after execute query update**

*Figure 1: Data after update*

## 4. QUERIES TO DELETE DATA WITH ILLUSTRATIONS OF FINAL RESULT

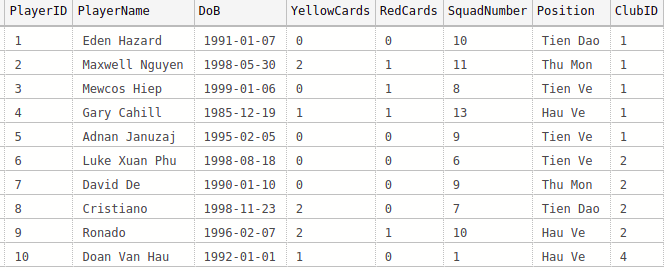
- During system management, if we want to delete any table information, we use the "DELETE" statement.

Example:

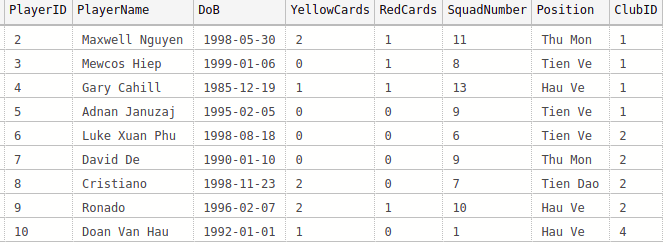
* User wants to delete player's information with id = 1, we write the following query:



*Figure 1: Example of query delete 1 record*

+ Use “SELECT” statement to show data before delete

*Figure 1: Data before delete*

+ Use “SELECT” statement to show data after execute query delete**

*Figure 1: Data after delete*

- To delete all data information of a certain table, we use the following query:

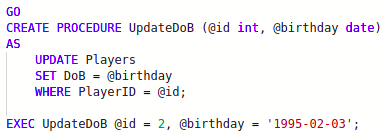


*Figure 1: Example of query delete all datas of 1 table*

## 5. ADVANCED QUERIES: STORED PROCEDURES, TRIGGERS, FUNCTIONS

**\* Stored procedure**

- Example 1: execute update a certain player's birthday

+ Create procedure then execute call procedure

*Figure 1: Stored procedure “UpdateDoB”*

* Result after call procedure

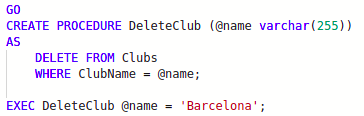


*Figure 1: Data before call procedure*

**

*Figure 1: Data after call procedure*

- Example 2: execute deletes the information of a certain club

+ Create procedure then execute call procedure

*Figure 1: Stored procedure “DeleteClub”*

* Result after call procedure



*Figure 1: Data before call procedure*

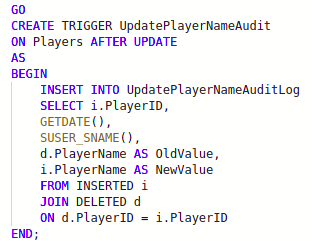
**

*Figure 1: Data after call procedure*

**\* Trigger**

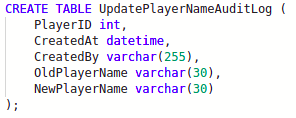
- Example 1: trigger to track name changes of Players table

+ Create 1 trigger



*Figure 1: Trigger “UpdatePlayerNameAudit”*

* Result after execute update name of any record in Players table



*Figure 1: Create table “UpdatePlayerNameAuditLog”*

**

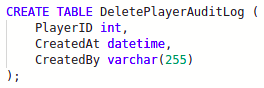
*Figure 1: Execute update a record in Players table*



*Figure 1: Data log table after execute update*

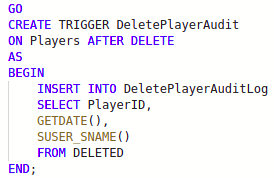
- Example 2: trigger to track the deletion of information from the Players table

* Create 1 trigger



*Figure 1: Trigger “DeletePlayerAudit”*

* Result after execute delete information of any record in Players table



*Figure 1: Create table “DeletePlayerAuditLog”*

**

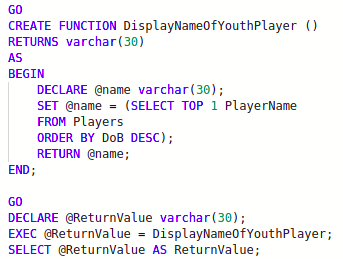
*Figure 1: Execute delete a record in Players table*



*Figure 1: Data log table after execute delete*

**\* Function**

- Example 1: display information of the youngest player

+ Create function then execute call function

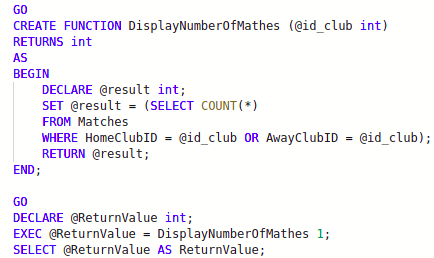
*Figure 1: Function “DisplayNameOfYouthPlayer”*

* Result after call function



*Figure 1: Data after call procedure*

- Example 2: display the number of matches the team with code = 1 has ever participated in

+ Create function then execute call function

*Figure 1: Function “DisplayNumberOfMathes”*

* Result after call function



*Figure 1: Data after call procedure*

# III. TEST THE SYSTEM

## 1. TEST PLAN

- In this test, we will use the data inserted in P2 to execute the test case to check if the system is correct.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case** | **Content** | **Times** | **Expected result** |
| 1 | Insert data into table | 1 | Run successful and display result correct |
| 2 | Update data in table | 1 | Run successful and display result correct |
| 3 | Delete data in table | 1 | Run successful and display result correct |
| 4 | Select to view table | 1 | Run successful and display result correct |
| 5 | Call stored procedure “UpdateDoB” | 1 | Run successful and display result correct |
| 6 | Call stored procedure “DeleteClub” | 1 | Run successful and display result correct |
| 7 | Check operator of trigger “UpdatePlayerNameAudit” | 1 | Run successful and display result correct |
| 8 | Check operator of trigger “DeletePlayerAudit” | 1 | Run successful and display result correct |
| 9 | Call function “DisplayNameOfYouthPlayer” | 1 | Run successful and display result correct |
| 10 | Call function “DisplayNumberOfMathes” | 1 | Run successful and display result correct |

## 2. TEST RESULT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case** | **SQL code** | **Expected result** | **Test result** | **Status** |
| 1 |  | In table Clubs, exist 10 records with data same as inserted data |  | Pass |
| 2 |  | In table Players, for PlayerID = 3, SquadNumber = 8 and DoB = '1999-01-06' |  | Pass |
| 3 |  | In table Players, doesn’t exist record contains PlayerID = 1 |  | Pass |
| 4 |  | Show all data in table Clubs, exist 10 records |  | Pass |
| 5 |  | In table Players, with record of PlayerID = 2, DoB will be updated to new value (DoB = ‘1995-02-03’) |  | Pass |
| 6 |  | In table Clubs, doesn’t exist record with ClubName = ‘Barcelona’ |  | Pass |
| 7 |  | 1 record will be inserted into “UpdatePlayerAuditLog” table |  | Pass |
| 8 |  | 1 record will be inserted into “DeletePlayerAuditLog” table |  | Pass |
| 9 |  | Return value = ‘Mewcos Hiep’ |  | Pass |
| 10 |  | Return value = 3 |  | Pass |

# IV. PRODUCE TECHNICAL AND USER DOCUMENTATION (P5)

## 1. TECHNICAL DOCUMENT

### 1.1 INTRODUCTION

- Purpose: The goal of this project is to solve the problem of the Football Club. This study focuses on the design of a database system. This topic's system is as follows: Create a database system that includes Football Players, Club Names, Matches, Statistical Data Chart, and Locations for easier and more effective management football game

Development will proceed from the first step of gathering information to the implementation of a complete system to meet everyone's needs. Schedule upgrades and maintenance to prepare for data system failures.

-Situation: Currently, there is an increase in the demand for watching football as entertainment on television. Because of the rapid development of the Team, the Match also encountered some difficulties. As a result, this project was created with the goal of making Team and Match management easier and more effective. Matches, in addition to Teams. Employees or customers want to watch the football game; the management system also assists the restaurant manager in compiling statistics on the issues that need to be resolved in developing the Match and better managing the Team.

### 1.2 USER REQUIREMENT

This section was developed with the goal of describing the product features that the database system can display to the user. All the following features will be available in the fully built Football Player, Match management system:

• As a manager, I want to

-show Footballers (All staff information must have a unique Player\_ID to locate those Players in the entire database.), Club name, Match, CHart Stats and football position so I can report specifics to the football team manager,

-Manage the team's work schedule and specific tasks to be able to organize them effectively.

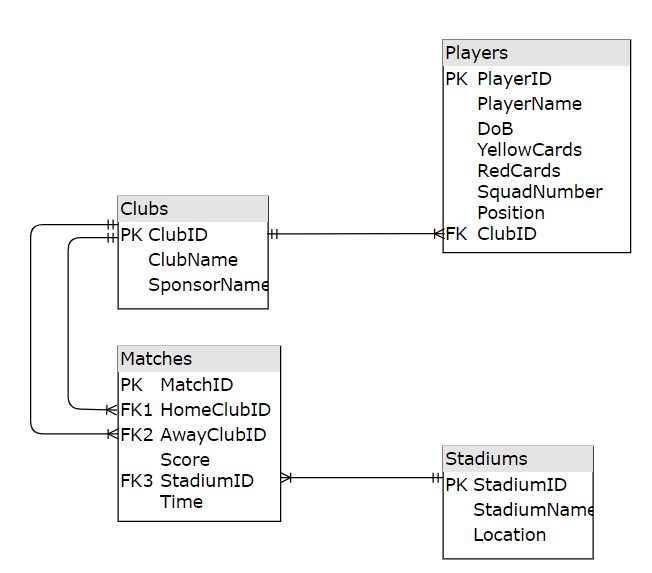
-add, remove and update data to suit customer needs and manage team for error checking

• As a football team,

-I want to see how many yellow card with red cards fouls

-check my football venue location to check if my information in the system is correct.

### 1.3 DIAGRAM



### 1.4 POSSIBLE AND REALISTIC IMPROVEMENT

Because the number of queries required will take a long time, the indexing of the system must be improved. Indexes should be used to improve performance.

Use extended stored procedures and triggers to process jobs quickly. Claims report forms should be created.

### 1.5 CURRENT VERSION NUMBER

SQL server 2022

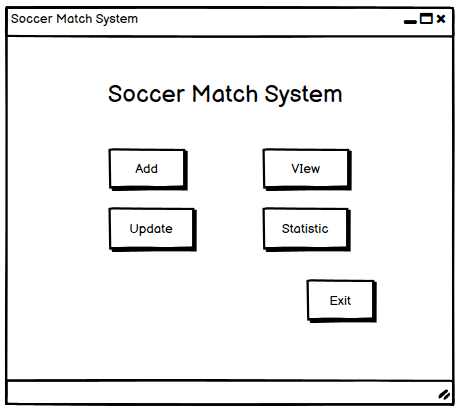
App.diagrams.net

### 1.6 DATE LAST CHANGED AND IMPLEMENTED

SQL server 2022: 10/7/2022

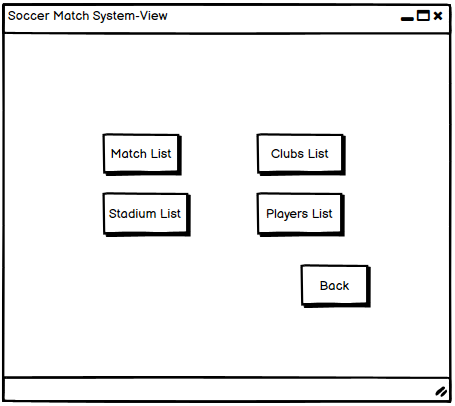
## 2. USER DOCUMENT

### Menu window

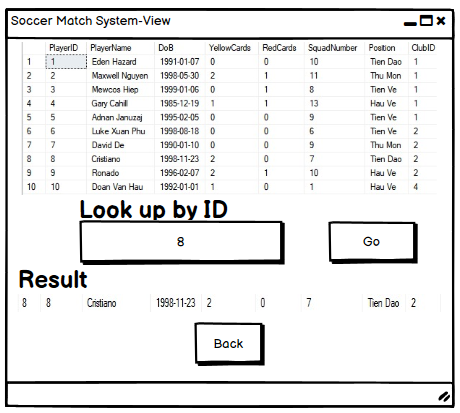


* 1: View data
* 2: Add new data
* 3: Update data
* 4: Statistic information
* • 5: Exit system

#### View window

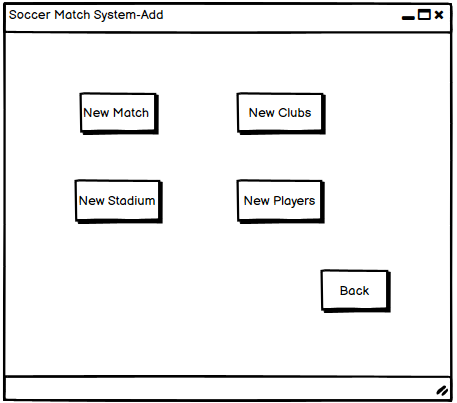


* 1: View Match List
* 2: View Clubs List
* 3: View Stadium List
* 4: View Players List
* 5: Back to Menu

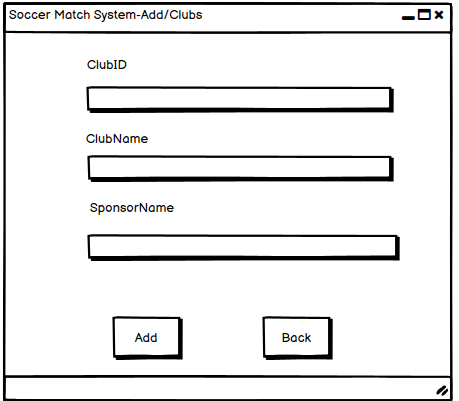


* 1: List of information the user wants to view
* 2: Input ID that user want to find out
* 3: Click Go to start find
* 4: Back to View menu

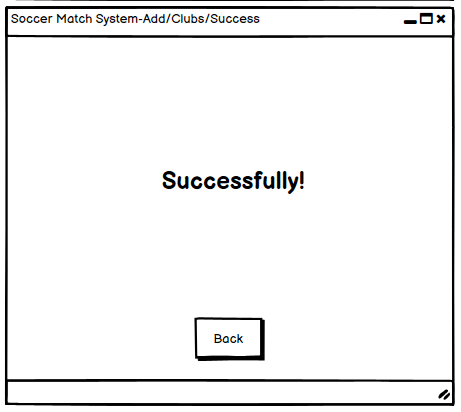
#### Add window



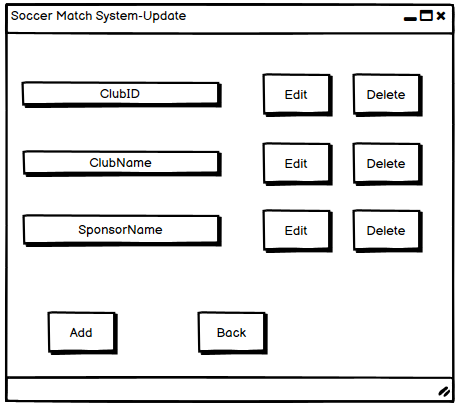
* 1: Add Match List
* 2: Add Clubs List
* 3: Add Stadium List
* 4: Add Players List
* 5: Back to main Menu



* 1: Input information
* 2: Click “Add” to finish ❖ Window when Add success:
* 3: Back to Add menu

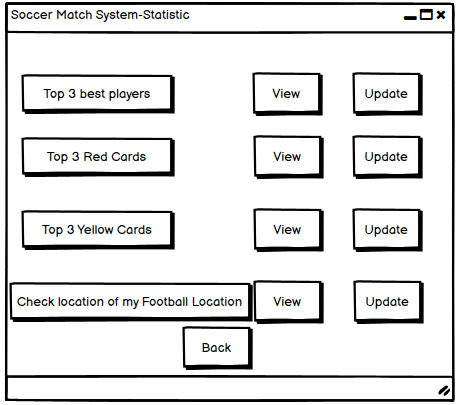


#### Update window



* 1: List of information that the Club wants to update
* 2: Click “Edit” to update data
* 3: Click “Delete” to delete data
* 4: Back to main menu

#### Statistic window



* 1: List of statistics that the user wants to view
* 2: Click “View” to view data
* 3: Click “Update” to update data
* 4: Back to main menu

# REFERENCE LIST