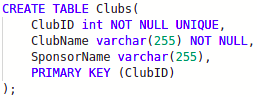
**2. OUTPUTS AND DATA VALIDATION  
2.1 QUERIES TO CREATE DATABASE AND TABLE**

- Khi bắt đầu tạo cơ sở dữ liệu cho hệ thống, việc đầu tiên cần làm là tạo 1 database để lưu lại thông tin các bảng

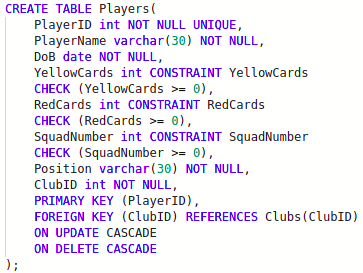


*Figure 1: Create database for system*

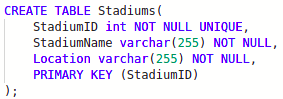
- Tiếp theo, chúng ta tạo các bảng theo như design ban đầu



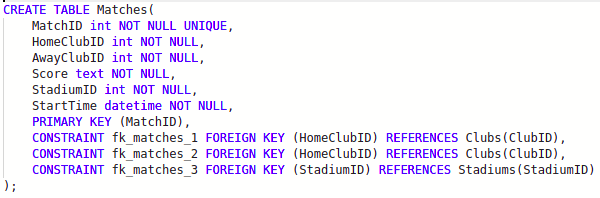
*Figure 1: Create Clubs table*



*Figure 1: Create Players table*



*Figure 1: Create Stadiums table*

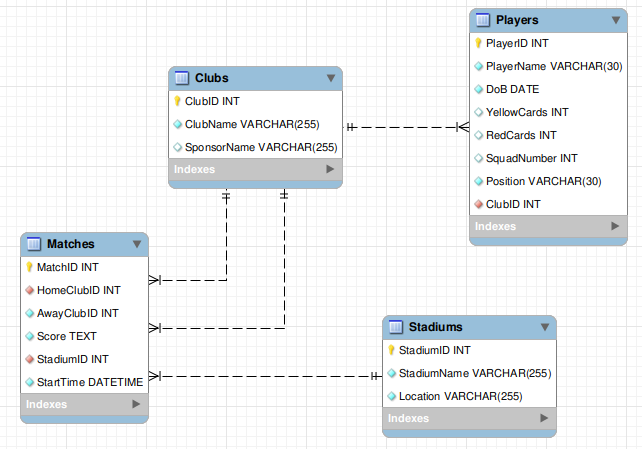


*Figure 1: Create Matches table*

- Các khóa chính và khóa ngoại sẽ được generated trong khi create table. After create tables, we sẽ được kết quả như hình bên dưới đây:



*Figure 1: Result of table creation*

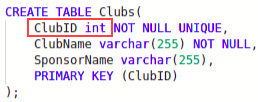


*Figure 1: Database diagram of system*

**2.2 DATA VALIDATION**

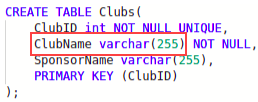
**\* Data type**

- 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)



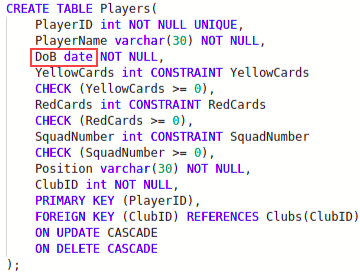
*Figure 1: Example of “int”**data type*

- 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



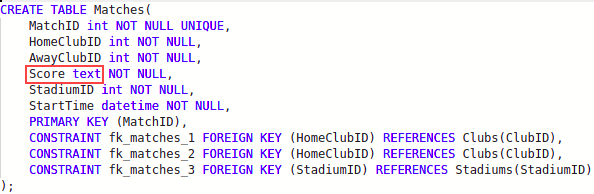
*Figure 1: Example of “*varchar*”**data type*

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



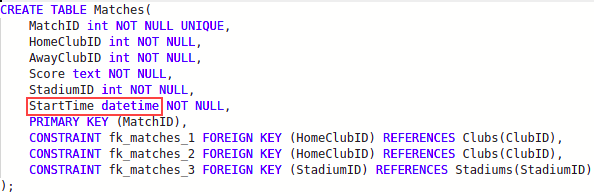
*Figure 1: Example of “*date*”**data type*

- 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

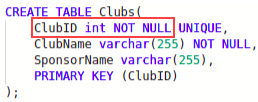
**

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

**\* Presence**

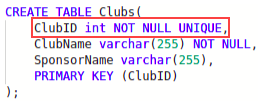
CONSTRAINT is used to define rules to allow or restrict what values can be stored in columns. The purpose of inducing constraints is to enforce the integrity of a database

- 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



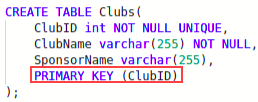
*Figure 1: Example of “NOT NULL”**constraint*

- 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

**

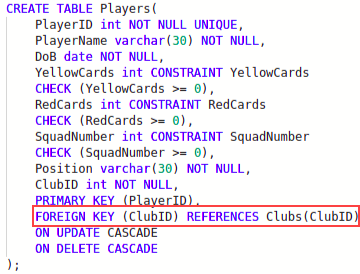
*Figure 1: Example of “UNIQUE”**constraint*

- 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



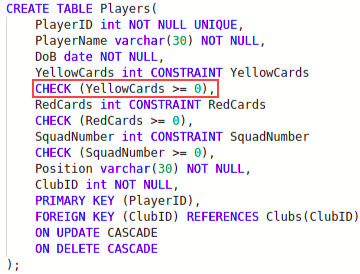
*Figure 1: Example of “PRIMARY KEY”**constraint*

- 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

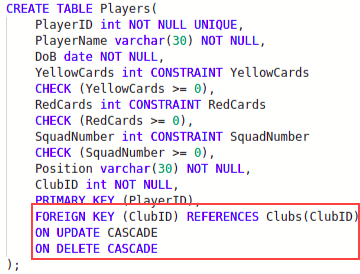
**

*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.



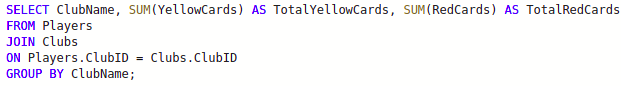
*Figure 1: Example of “CASCADE” data integrity*

Với câu sql tạo bảng bên trên, chúng ta đã define ON UPDATE CASCADE và ON DELETE CASCADE cho khóa ngoại của bảng Players. Khi chúng ta update ClubID của bảng Clubs thì data của bảng Players (nghĩa là ClubID) cũng sẽ thay đổi theo. Còn nếu chúng ta xóa thông tin chứa ClubID trong bảng Clubs thì data của bảng Players sẽ xóa toàn bộ tất cả các record có liên quan đến ClubID mà đã bị xóa

- 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.

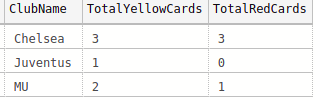
Here are the different types of the JOINs in SQL:

* (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*

Câu query này dùng để đếm số thẻ vàng và thẻ đỏ của tất cả các cầu thủ trong 1 câu lạc bộ. Chúng ta cần phải sử dụng “JOIN” statement để liên kết 2 bảng Players và Clubs lại với nhau dựa trên ClubID. Tiếp theo, nhóm chúng lại với nhau theo từng ClubName thông qua “GROUP BY” statement. Sau đó, cần sử dụng “SELECT” statement để lấy ClubName và tính toán tổng thẻ vàng, thẻ đỏ.

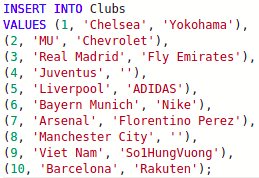


*Figure 1: Result of “Querying Across Multiple Tables”*

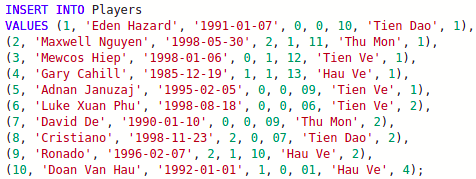
**II. PRODUCE QUERIES**

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

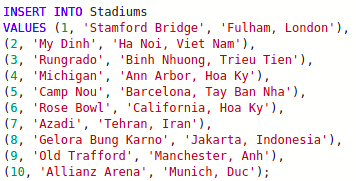
- Sau khi tạo bảng thành công, chúng ta cần insert data information vào tables



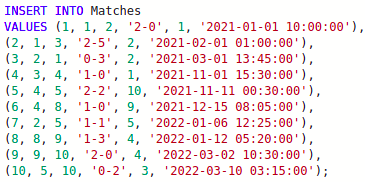
*Figure 1: Insert data into “Clubs” table*

**

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

**

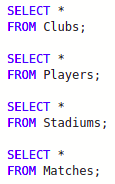
*Figure 1: Insert data into “Stadiums” table*

**

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

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

- Sau khi thực hiện insert data vào cơ sở dữ liệu, chúng ta có thể kiểm tra thông tin đã được insert vào database bằng cách sử dụng “SELECT” statement

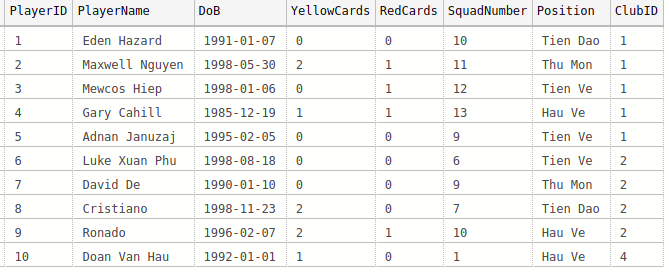


*Figure 1: Select data of all tables*

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



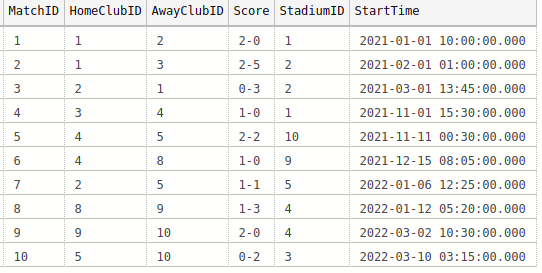
*Figure 1: Data information of Clubs table*

**

*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**

- Trong suốt quá trình quản lý hệ thống, nếu thông tin được insert vào DB không đúng, cần sửa lại thông tin cho đúng hoặc muốn cập nhật lại thông tin sau khi đã insert vào thì chúng ta sử dụng “UPDATE” statement

Example:

* User muốn update một vài thông tin của cầu thủ có id = 3 như mang áo số 8, sinh ngày “1999-01-06”, chúng ta viết câu query như sau:



*Figure 1: Example of query update data*

* Sử dụng “SELECT” statement để show data before update



*Figure 1: Data before update*

* Sử dụng “SELECT” statement để show data after execute query update

**

*Figure 1: Data after update*

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

- Trong quá trình quản lý hệ thống, nếu muốn xóa bất kỳ information của bảng nào thì chúng ta sử dụng “DELETE” statement

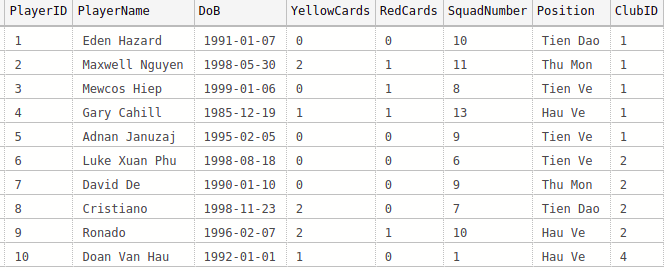
Example:

* User muốn xóa thông tin của cầu thủ có id = 1, chúng ta viết câu query như sau:



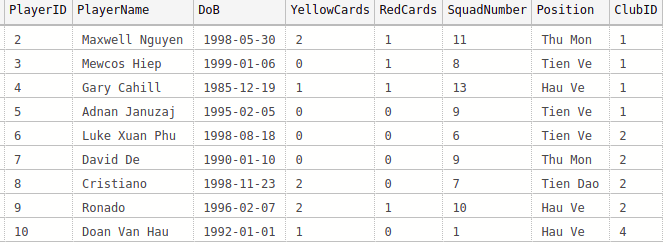
*Figure 1: Example of query delete 1 record*

* Sử dụng “SELECT” statement để show data before *delete*



*Figure 1: Data before delete*

* Sử dụng “SELECT” statement để show data after execute query *delete*

**

*Figure 1: Data after delete*

- Để xóa toàn bộ data information của 1 bảng nào đó, chúng ta sử dụng query sau:



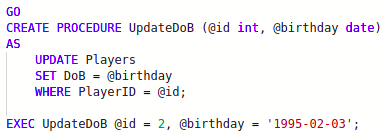
*Figure 1: Example of query delete all datas of 1 table*

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

**\* Stored procedure**

- Example 1: execute update ngày sinh của một cầu thủ nào đó

* Tạo procedure then execute call procedure



*Figure 1: Stored procedure “UpdateDoB”*

* Result after call procedure



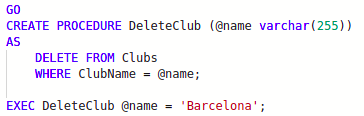
*Figure 1: Data before call procedure*

**

*Figure 1: Data after call procedure*

- Example 2: execute xóa thông tin của một câu lạc bộ nào đó

* Tạo procedure then execute call procedure



*Figure 1: Stored procedure “DeleteClub”*

* Result after call procedure



*Figure 1: Data before call procedure*

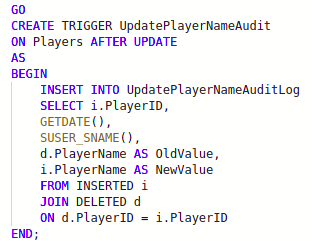
**

*Figure 1: Data after call procedure*

**\* Trigger**

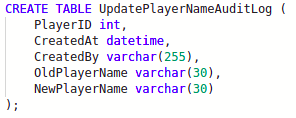
- Example 1: trigger to theo dõi việc sửa name của Players table

* Tạo 1 trigger



*Figure 1: Trigger “UpdatePlayerNameAudit”*

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



*Figure 1: Create table “UpdatePlayerNameAuditLog”*

**

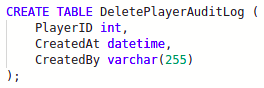
*Figure 1: Execute update a record in Players table*



*Figure 1: Data log table after execute update*

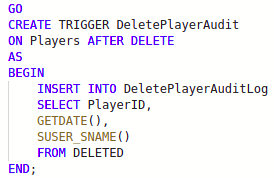
- Example 2: trigger to theo dõi việc xóa thông tin của Players table

* Tạo 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*

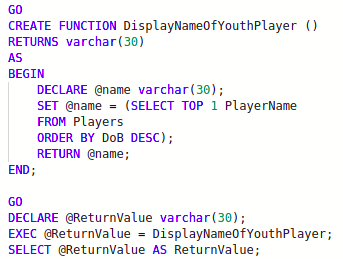


*Figure 1: Data log table after execute delete*

**\* Function**

- Example 1: display information cầu thủ trẻ tuổi nhất

* Tạo function then execute call function



*Figure 1: Function “DisplayNameOfYouthPlayer”*

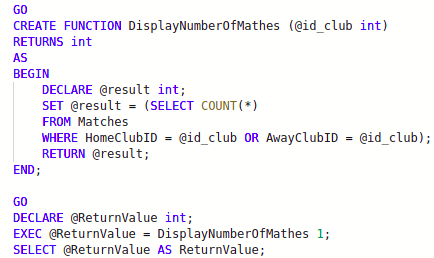
* Result after call function



*Figure 1: Data after call procedure*

- Example 2: display số trận đấu của đội có mã = 1 đã từng tham gia

* Tạo 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**

- Trong phần test này, chúng ta sẽ sử dụng data đã được insert ở P2 to execute test case to check hệ thống đã đúng chưa

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

**2. TEST RESULT**

| **Test case** | **SQL code** | **Expected result** | **Test result** | **Status** |
| --- | --- | --- | --- | --- |
| 1 |  | Trong bảng Clubs, exist 10 records with data same as inserted data |  | Pass |
| 2 |  | Trong bảng Players, đối với PlayerID = 3 thì SquadNumber = 8 and DoB = ‘1999-01-06’ |  | Pass |
| 3 |  | Trong bảng 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 |