NBA League

universidade de aveiro

Base de Dados 22-23

André Oliveira – 107637
Duarte Cruz – 107359

P4– G1

Índice

Introdução

A escolha do nosso projeto foi motivada pelo interesse em comum dos membros do grupo na NBA. A aplicação permitirá o acompanhamento da temporada atual da liga.

A interface foi desenvolvida para o uso administrativo para a manipulação dos diversos dados, mas na perspetiva do utilizador, estes poderiam acompanhar a classificação geral da temporada, informações sobre equipas específicas e seus jogadores assim como, sobre os treinadores. Para além disto, poderão até mesmo consultar os bilhetes restantes para cada jogo.

Além disso, a nossa aplicação oferecerá informações sobre resultados de jogos, estatísticas dos jogadores, etc. Isto permitirá aos usuários se manterem atualizados sobre as últimas novidades da NBA numa única plataforma.

Futuramente, a nossa plataforma poderia permitir que os usuários criem seus próprios perfis e se conectem com outros fãs da NBA, compartilhando opiniões e participando de discussões sobre a liga, tornando a experiência mais interativa e social para os usuários.

Para desenvolver a nossa aplicação, planeamos recolher dados de diversas fontes, incluindo *websites* das diversas equipas.

Em resumo, a nossa aplicação, por agora, forneceria uma solução completa para os fãs da NBA, oferecendo informações atualizadas e relevantes e a possibilidade de busca de informações de bilhetes para jogos, tudo em um só lugar.

Na realização deste projeto foi utilizada uma pasta partilhado no OneDrive como plataforma de organização. O projeto também se encontra disponível no GitHub, mas sublinhamos que os *commits* não correspondem à participação de cada elemento do grupo.

Estrutura da Pasta do Projeto

Dentro do repositório Projeto-BD e do ficheiro .zip é possível encontrar 3 imagens (diagrama DER, diagrama ER e diagrama ER do SGBD), uma pasta SQL com ficheiros .sql divididos de acordo com o tipo de queries (DDL.sql, DML.sql, INDEXES.sql, STORED_PROCEDURES.sql, TRIGGERS.sql, UDF.sql e VIEWS.sql), a pasta Proposta com os ficheiros entregues anteriormente na proposta do projeto, o pdf Apresentação que corresponde ao powerpoint usado na apresentação do projeto na aula e os ficheiros relativos à interface gráfica (Pasta Projeto e o ficheiro Projeto.sln inicia a interface).

- **Pasta SQL:**
    - **DDL.sql** – *Script SQL* que contém a definição de todas as tabelas necessárias, assim como definições de chaves primárias e estrangeiras e algumas verificações para os atributos de cada tabela;
    - **DML.sql** – *Script SQL* que contém os dados a ser inseridos em cada tabela;
    - **INDEXES.sql** – *Script SQL* que contém os *Indexes*;
    - **STORED_PROCEDURES.sql** – *Script SQL* que contém os *Stored Procedures*;
    - **TRIGGERS.sql** – *Script SQL* que contém os *Triggers*;
    - **UDF.sql** – *Script SQL* que contém as *User Defined Functions*;
    - **VIEWS.sql** – *Script SQL* que contém as *Views*.

Nota: A cada execução de um ficheiro .sql, é tudo refeito, não sendo necessário apagar nada antes.

Requisitos Funcionais

Entidade	Funcionalidades
Utilizador	• Procurar informações sobre cada equipa, sobre os respetivos treinadores das equipas e sobre os jogadores • Procurar informações sobre jogos, assim como os bilhetes para os mesmos que ainda não ocorreram • Acompanhar a classificação da liga atual • Procurar estatísticas das equipas e de respetivos jogadores (média e de cada jogo)
Administrador	• Inserir/alterar/remover dados sobre jogos, treinadores, jogadores e equipas

Entidades

Team – Representa uma equipa participante da liga. Possui um ID, um nome, uma conferência, um ano de fundação e é de uma cidade. Esta, tem vários jogadores, um treinador e um presidente.

Person – Representa a instância mais geral do treinador e do jogador. Possui um nome, uma idade e um número de cartão de cidadão. Esta pode ter um contrato associado.

Player – Representa um jogador de uma equipa. Possui um nome, uma idade, um número de cartão de cidadão, uma altura, um peso, uma posição de jogo e um número de equipamento. Este possui uma estatística média.

Coach – Representa um treinador de uma equipa. Possui um nome, uma idade um número de cartão de cidadão.

Contract – Representa um contrato assinado por um treinador ou jogador ou pessoa que será presidente de uma equipa. Possui um ID, descrição, salário, data de início e data de fim.

Stadium – Representa um estádio. Possui um ID, localização, capacidade e nome. Este pertence a uma equipa e nele são realizados vários jogos.

Average Individual Numbers – Representa a média da estatística individual de um jogador. Esta possui pontos, abafos, roubos de bola, assistências, ressaltos, percentagem de triplo e percentagem de lançamentos de campo.

Game – Representa um jogo entre duas equipas (visitada e visitante). Possui um ID, hora, data e pontos da equipa visitada e da visitante. Este é realizado num estádio.

Ticket – Representa os bilhetes que estão a ser vendidos por uma equipa para um determinado jogo. Possui um tipo, um preço e um número de bilhetes restantes.

Diagramas

Diagrama Entidade-Relacionamento

Abaixo está ilustrado o diagrama entidade-relacionamento correspondente às entidades referidas acima e relações entre as mesmas.

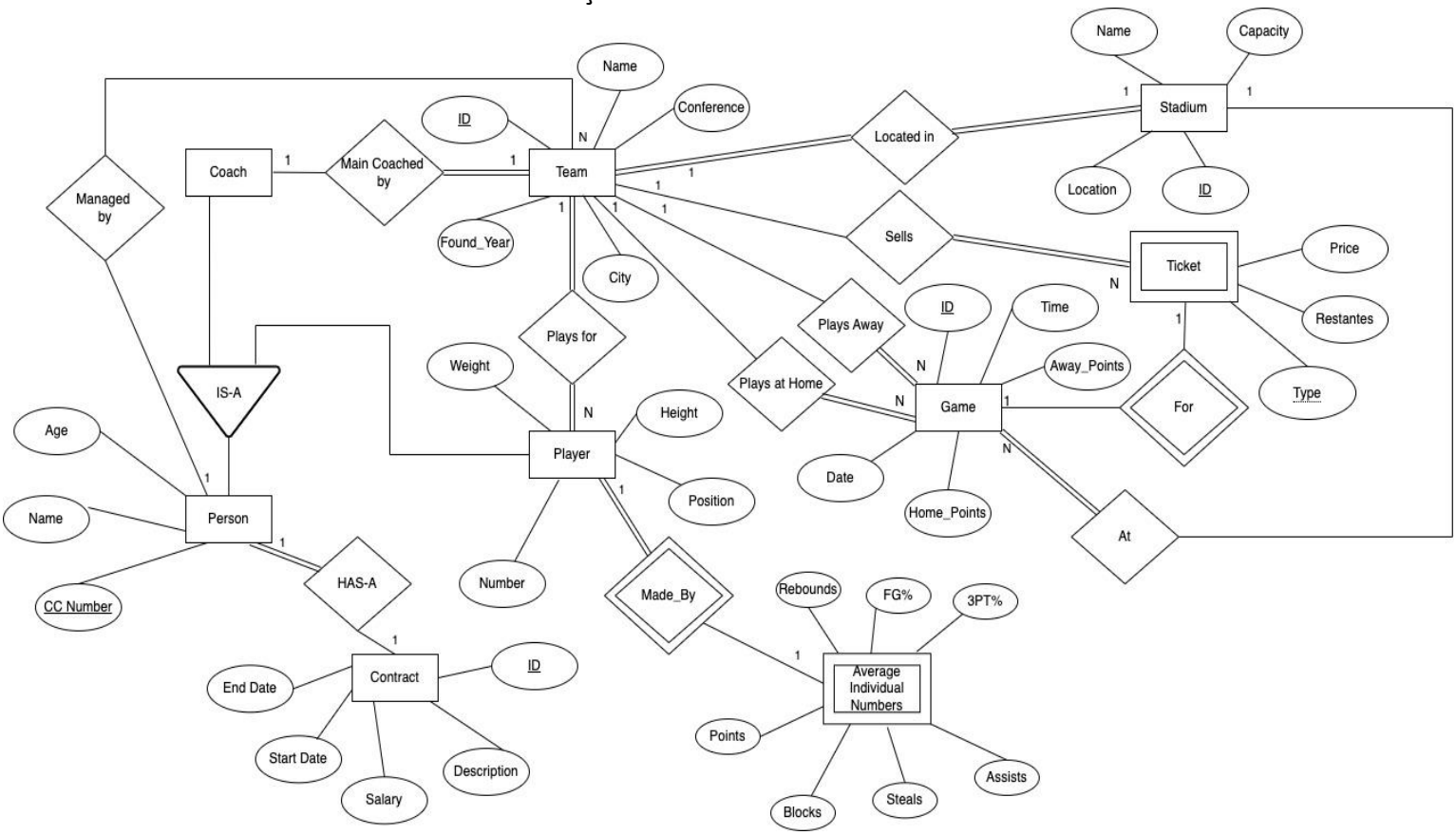

Figura 1 | Digrama Entidade-Relacionamento

Esquema Relacional

Abaixo está ilustrado o esquema relacional de acordo com o diagrama entidade-relacionamento desenvolvido.

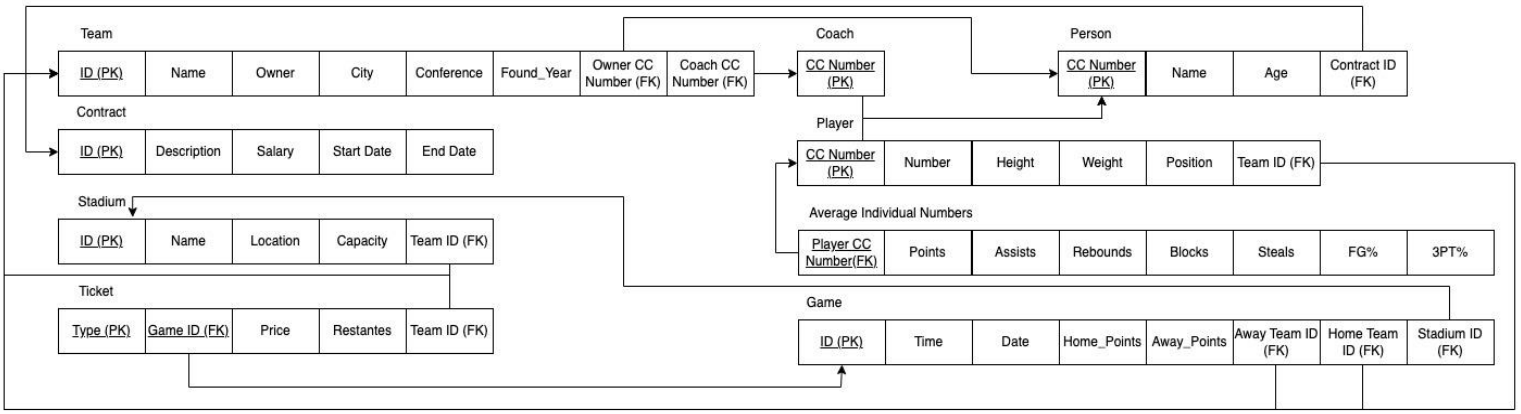

Figura 2 | Esquema Relacional

Esquema Relacional (SGDB)

Abaixo está ilustrado o esquema relacional criado pelo SGBD utilizado.

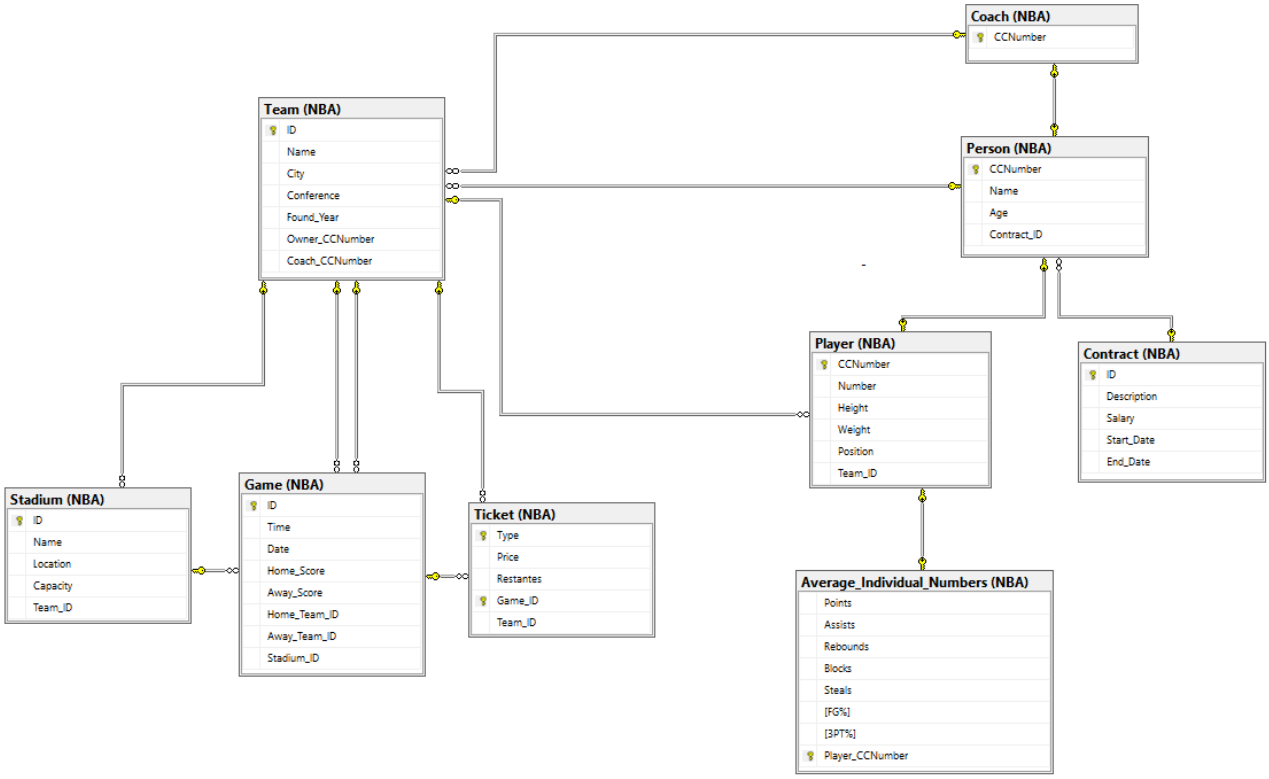

Figura 3 | Esquema Relacional (SGBD)

Queries SQL

Data Definition Language (DDL)

A DDL é uma parte fundamental da linguagem SQL (Structured Query Language) e permite definir a estrutura dos dados dos objetos a ser guardados na base de dados.

Abaixo é apresentado o código que leva à geração das tabelas necessárias ao nosso projeto e nele verificamos que já existem diversas verificações nos dados de cada tabela.

```
create table NBA.[Contract] (
    ID                  int             not null,
    [Description]       varchar(50)     not null,
    Salary              float           not null        check(Salary > 0),
    [Start_Date]        date            not null,
    End_Date            date            not null,

    check (End_Date > [Start_Date]),
    primary key (ID),
    unique ([Description])
);
create table NBA.Person (
    CCNumber            int             not null        check(len(CCNumber) = 8),
    [Name]              varchar(50)     not null,
    Age                 int             not null        check(Age > 0),
    Contract_ID         int,

    primary key (CCNumber)
);
create table NBA.Team (
    ID              int             not null,
    [Name]          varchar(50)     not null,
    City            varchar(50)     not null,
    Conference      varchar(50)     not null,
    Found_Year      int             not null        check(Found_Year > 0),
    Owner_CCNumber  int,
    Coach_CCNumber  int,
    disabled        bit             default 0,

    primary key (ID),
    unique ([Name])
);
create table NBA.Stadium (
    ID                  int             not null,
    [Name]              varchar(50)     not null,
    [Location]          varchar(50)     not null,
    Capacity            int             not null        check(Capacity > 0),
    Team_ID             int             not null,

    primary key (ID),
    unique ([Name])
);
create table NBA.Ticket (
    [Type]              varchar(30)     not null,
    Price               decimal(10,2)   not null        check(Price > 0),
    Restantes           int                             check(Restantes >= 0),
    Game_ID             int             not null,
    Team_ID             int             not null
);
create table NBA.Coach (
    CCNumber            int             not null        check(len(CCNumber) = 8),

    primary key (CCNumber)
);



create table NBA.Player (
```

```
    CCNumber                int                 not null        check(len(CCNumber) = 8),
    [Number]                int                 not null        check([Number] >= 0),
    Height                  varchar(5)          not null        check(Height like '[0-9]-[0-9]'),
    [Weight]                float               not null        check([Weight] > 0),
    Position                varchar(20)         not null,
    Team_ID                 int,

    primary key (CCNumber)
);
create table NBA.Average_Individual_Numbers (
    Points                  decimal(10, 2)                      check([Points] >= 0),
    Assists                 decimal(10, 2)                      check([Assists] >= 0),
    Rebounds                decimal(10, 2)                      check([Rebounds] >= 0),
    Blocks                  decimal(10, 2)                      check([Blocks] >= 0),
    Steals                  decimal(10, 2)                      check([Steals] >= 0),
    [FG%]                   decimal(10, 2)                      check([FG%] >= 0 and [FG%] <= 100),
    [3PT%]                  decimal(10, 2)                      check([3PT%] >= 0 and [3PT%] <= 100),
    Player_CCNumber         int                                 not null,
);
create table NBA.Game (
    ID                      int                 not null,
    [Time]                  time                not null,
    [Date]                  date                not null,
    Home_Score              int                 check(Home_Score > 0),
    Away_Score              int                 check(Away_Score > 0),
    Home_Team_ID            int                 not null,
    Away_Team_ID            int                 not null,
    Stadium_ID              int                 not null,

    check(Home_Score != Away_Score),
    primary key (ID),
);

alter table NBA.Player add constraint PlayerIDFK foreign key (CCNumber) references
NBA.Person(CCNumber);
alter table NBA.Player add constraint TeamFK3 foreign key (Team_ID) references NBA.Team(ID)
    on update cascade on delete set null;


alter table NBA.Coach add constraint CoachIDFK foreign key (CCNumber) references NBA.Person(CCNumber);


alter table NBA.Team add constraint CoachFK foreign key (Coach_CCNumber) references NBA.Coach(CCNumber)
    on update cascade on delete set null;
alter table NBA.Team add constraint OwnerFK foreign key (Owner_CCNumber) references
NBA.Person(CCNumber)
    on update cascade on delete set null;


alter table NBA.Stadium add constraint TeamFK1 foreign key (Team_ID) references NBA.Team(ID)
    on update cascade on delete cascade;


alter table NBA.Ticket add constraint GameFK foreign key (Game_ID) references NBA.Game(ID)
    on update cascade on delete cascade;
alter table NBA.Ticket add constraint TeamFK2 foreign key (Team_ID) references NBA.Team(ID)
    on update cascade on delete cascade;
alter table NBA.Ticket add primary key ([Type], Game_ID);


alter table NBA.Person add constraint ContractFK foreign key (Contract_ID) references
NBA.[Contract](ID)
    on update cascade on delete set null;


alter table NBA.Average_Individual_Numbers add constraint PlayerFK foreign key (Player_CCNumber)
references NBA.Player(CCNumber)
    on update cascade on delete cascade;
alter table NBA.Average_Individual_Numbers add primary key (Player_CCNumber);


alter table NBA.Game add constraint HomeTeamFK foreign key (Home_Team_ID) references NBA.Team(ID);
alter table NBA.Game add constraint AwayTeamFK foreign key (Away_Team_ID) references NBA.Team(ID);
alter table NBA.Game add constraint StadiumFK foreign key (Stadium_ID) references NBA.Stadium(ID);
```

Data Manipulation Language (DML)

A DML também é uma parte essencial da linguagem SQL e desempenha um papel crucial inserção de dados numa base de dados.

A partir da mesma, inserimos diversos dados nas tabelas criadas acima. Abaixo são apresentados exemplos da inserção dos dados.

```
insert into NBA.Team (ID, [Name], City, Conference, Found_Year, Owner_CCNumber,
Coach_CCNumber) values
(1, 'Los Angeles Lakers', 'Los Angeles', 'Western', 1947, 10000111, 10000101),
(2, 'Boston Celtics', 'Boston', 'Eastern', 1946, 10000112, 10000102),
(3, 'Golden State Warriors', 'San Francisco', 'Western', 1946, 10000113, 10000103),
(4, 'New York Knicks', 'New York', 'Eastern', 1946, 10000114, 10000104),
(5, 'Chicago Bulls', 'Chicago', 'Eastern', 1966, 10000115, 10000105),
(6, 'Miami Heat', 'Miami', 'Eastern', 1988, 10000116, 10000106),
(7, 'Dallas Mavericks', 'Dallas', 'Western', 1980, 10000117, 10000107),
(8, 'San Antonio Spurs', 'San Antonio', 'Western', 1967, 10000118, 10000108),
(9, 'Houston Rockets', 'Houston', 'Western', 1967, 10000119, 10000109),
(10, 'Portland Trail Blazers', 'Portland', 'Western', 1970, 10000120, 10000110);

insert into NBA.Stadium (ID, [Name], [Location], Capacity, Team_ID) values
(1, 'Staples Center', 'Los Angeles', 19060, 1),
(2, 'TD Garden', 'Boston', 18624, 2),
(3, 'Chase Center', 'San Francisco', 18064, 3),
(4, 'Madison Square Garden', 'New York', 19812, 4),
(5, 'United Center', 'Chicago', 20917, 5),
(6, 'American Airlines Arena', 'Miami', 19600, 6),
(7, 'American Airlines Center', 'Dallas', 19200, 7),
(8, 'AT&T Center', 'San Antonio', 18418, 8),
(9, 'Toyota Center', 'Houston', 18055, 9),
(10, 'Moda Center', 'Portland', 19441, 10);

insert into NBA.Ticket ([Type], Price, Restantes, Game_ID, Team_ID) values
('Regular', 150, 1000, 61, 1),
('VIP', 500, 100, 61, 1),
('Regular', 150, 1000, 62, 3),
('VIP', 500, 100, 62, 3),
('Regular', 150, 1000, 63, 1),
('VIP', 500, 100, 63, 1),
('Regular', 150, 1000, 64, 2),
('VIP', 500, 100, 64, 2),
('Regular', 150, 1000, 65, 5),
('VIP', 500, 100, 65, 5),
('Regular', 150, 1000, 66, 6),
('VIP', 500, 100, 66, 6),
('Regular', 150, 1000, 67, 7),
('VIP', 500, 100, 67, 7),
('Regular', 150, 1000, 68, 8),
('VIP', 500, 100, 68, 8),
('Regular', 150, 1000, 69, 9),
('VIP', 500, 100, 69, 9),
('Regular', 150, 1000, 70, 10),
('VIP', 500, 100, 70, 10);
```

Views

As Views são uma poderosa ferramenta do SQL que permitem criar consultas personalizadas e armazená-las como objetos virtuais numa base de dados. Elas oferecem uma grande vantagem na facilidade do acesso e gerenciamento dos dados.

Neste projeto, estas foram bastante usadas, principalmente para facilitar a apresentação dos dados na interface, mas também em User Definition Functions (UDF's) e Stored Procedures.

Abaixo é apresentado o código que leva à criação destas Views.

```
-- Pessoa e jogador
drop view IF EXISTS NBA.PersonPlayer
go
create view NBA.PersonPlayer as
	select Pe.CCNumber, Pe.[Name], Pe.Age, Pe.Contract_ID, Pl.[Number], Pl.Height,
Pl.[Weight], Pl.Position, Pl.Team_ID, T.[Name] as TeamName
	from ((NBA.Player as Pl join NBA.Person as Pe on Pl.CCNumber = Pe.CCNumber) inner join
NBA.Team as T on Pl.Team_ID = T.ID);
go

-- Pessoa e treinador
drop view IF EXISTS NBA.PersonCoach
go
create view NBA.PersonCoach as
	select Pe.CCNumber, Pe.[Name], Pe.Age, Pe.Contract_ID
	from (NBA.Coach as Co join NBA.Person as Pe on Co.CCNumber = Pe.CCNumber);
go

-- Jogadores com contrato
drop view IF EXISTS NBA.PlayersWithContract
go
create view NBA.PlayersWithContract as
	select Pl.CCNumber, Pl.[Name], Pl.Age, Pl.[Number], Pl.Height, Pl.[Weight], Pl.Position,
Pl.Team_ID, C.ID, C.[Description], C.Salary,C.[Start_Date],C.End_Date
	from (NBA.PersonPlayer as Pl join NBA.[Contract] as C on Pl.Contract_ID = C.ID)
	where C.End_Date > getdate();
go

-- Jogadores sem contrato
drop view IF EXISTS NBA.PlayersWithoutContract
go
create view NBA.PlayersWithoutContract as
	select Pl.CCNumber, Pl.[Name], Pl.Age, Pl.[Number], Pl.Height, Pl.[Weight], Pl.Position,
Pl.Team_ID, C.ID, C.[Description],C.Salary,C.[Start_Date],C.End_Date
	from (NBA.PersonPlayer as Pl join NBA.[Contract] as C on Pl.Contract_ID = C.ID)
	where C.End_Date < getdate();
go

-- Treinadores com contrato
drop view IF EXISTS NBA.CoachesWithContract
go
create view NBA.CoachesWithContract as
	select Co.CCNumber, Co.[Name], Co.Age, C.ID,C.[Description], C.Salary ,C.[Start_Date],
C.End_Date
	from (NBA.PersonCoach as Co join NBA.[Contract] as C on Co.Contract_ID = C.ID)
	where C.End_Date > getdate();
go

-- Treinadores sem contrato
drop view IF EXISTS NBA.CoachesWithoutContract
go
create view NBA.CoachesWithoutContract as
	select Co.CCNumber, Co.[Name], Co.Age, C.ID ,C.[Description], C.Salary ,C.[Start_Date],
C.End_Date
	from (NBA.PersonCoach as Co join NBA.[Contract] as C on Co.Contract_ID = C.ID)
	where C.End_Date < getdate();
go
```

```
-- Equipas
drop view IF EXISTS NBA.TeamCoachOwner
go
create view NBA.TeamCoachOwner as
        select T.ID, T.[Name], T.City, T.Conference, T.Found_Year, C.[Name] as CoachName,
P.[Name] as OwnerName, C.CCNumber as CoachCCNumber, P.CCNumber as OwnerCCNumber
        from ((NBA.Team as T left outer join NBA.PersonCoach as C on T.Coach_CCNumber =
C.CCNumber) join NBA.Person as P on T.Owner_CCNumber = P.CCNumber)
        where T.disabled = 0;
go

-- Jogos com nome das equipas e pavlhão
drop view IF EXISTS NBA.GamesTeamsStadium
go
create view NBA.GamesTeamsStadium as
        select G.ID, G.[Time], G.[Date], G.Home_Score, G.Away_Score, T1.ID as HomeTeamID,
T1.[Name] as HomeTeamName, T2.ID as AwayTeamID, T2.[Name] as AwayTeamName, S.[Name] as
StadiumName, S.ID as StadiumID
        from ((NBA.Game as G inner join NBA.Team as T1 on G.Home_Team_ID = T1.ID) inner join
NBA.Team as T2 on G.Away_Team_ID = T2.ID) inner join NBA.Stadium as S on G.Stadium_ID = S.ID
go

-- Jogos com resultado
drop view IF EXISTS NBA.GamesWithResult
go
create view NBA.GamesWithResult as
        select G.ID, G.[Time], G.[Date], G.Home_Score, G.Away_Score, T1.[Name] as HomeTeamName,
T2.[Name] as AwayTeamName, S.[Name] as StadiumName
        from ((NBA.Game as G inner join NBA.Team as T1 on G.Home_Team_ID = T1.ID) inner join
NBA.Team as T2 on G.Away_Team_ID = T2.ID) inner join NBA.Stadium as S on G.Stadium_ID = S.ID
        where G.Home_Score is not null
go

-- Jogos com resultado
drop view IF EXISTS NBA.GamesWithoutResult
go
create view NBA.GamesWithoutResult as
        select G.ID, G.[Time], G.[Date], G.Home_Score, G.Away_Score, T1.[Name] as HomeTeamName,
T2.[Name] as AwayTeamName, S.[Name] as StadiumName
        from ((NBA.Game as G inner join NBA.Team as T1 on G.Home_Team_ID = T1.ID) inner join
NBA.Team as T2 on G.Away_Team_ID = T2.ID) inner join NBA.Stadium as S on G.Stadium_ID = S.ID
        where G.Home_Score is null
go
```

Indexes

Os Indexes desempenham um papel fundamental no desempenho e na eficiência de procura de dados em bases de dados. Eles são estruturas de dados auxiliares que aceleram o processo de pesquisa de informações. Com isto, concluímos que usar indexes leva a um aumento do desempenho e eficiência na busca de dados pela base de dados.

No nosso projeto criámos indexes com base nos campos das tabelas que são frequentemente utilizados na busca, filtragem e entre outros processos. Assim, apresentamos abaixo os indexes criados.

```
-- Criação de indexes na tabela NBA.[Contract]
create index seacrhContractStartDate on NBA.[Contract] ([Start_Date]);
create index searchContractEndDate on NBA.[Contract] ([End_Date]);

-- Criação de indexes na tabela NBA.Person
create index searchPersonName on NBA.Person ([Name]);

-- Criação de indexes na tabela NBA.Team
create index seacrhTeamName on NBA.Team ([Name]) where disabled = 0;

-- Criação de indexes na tabela NBA.Player
create index seacrhPlayerTeam on NBA.Player (Team_ID);

-- Criação de indexes na tabela NBA.Game
create index seacrhGameHomeTeamScore on NBA.Game (Home_Score);
create index seacrhGameAwayTeamScore on NBA.Game (Away_Score);
```

User Definition Functions (UDF's)

As UDF's são uma parte importante do SQL que permitem estender a funcionalidade da linguagem, criando funções personalizadas. Elas oferecem uma série de vantagens e utilidades que ajudam a simplificar o desenvolvimento de consultas complexas e a melhorar a reutilização de código.

Neste projeto, estas foram essencialmente usadas para o retorno de dados. Com isto, quer-se dizer que não foram usadas para, por exemplo, a adição, alteração ou exclusão de dados da base de dados.

Abaixo é apresentado o código das diferentes UDF's, tendo cada uma um comentário com a sua finalidade.

```
-- Função com os filtro de equipa, contrato e posição dos jogadores
drop function IF EXISTS NBA.filtrarJogadoresPorEquipaEContratoEPosicao;
go
create function NBA.filtrarJogadoresPorEquipaEContratoEPosicao(@equipa varchar(50), @contrato
varchar(3), @posicao varchar(30)) returns table
as
return (
    select P.CCNumber,P.[Name],P.Age,P.Contract_ID,P.Number,P.Height,P.[Weight],P.Position,P.Team_ID,
T.[Name] as TeamName
    from NBA.PersonPlayer as P join NBA.Team as T on P.Team_ID = T.ID
    where
        (T.[Name] = @equipa or @equipa is null) and (
            (@contrato = 'Sim' and P.CCNumber in (select CCNumber from NBA.PlayersWithContract))
            OR
            (@contrato = 'Nao' and P.CCNumber in (select CCNumber from NBA.PlayersWithoutContract))
            OR
            (@contrato is null)
        ) and ( P.Position = @posicao or @posicao is null)
);
go

-- Função com filtro de contrato dos treinadores
drop function IF EXISTS NBA.filtrarTreinadoresPorContrato;
go
create function NBA.filtrarTreinadoresPorContrato(@contrato varchar(3)) returns table
as
return (
    select CCNumber, [Name], Age, Contract_ID
    from NBA.PersonCoach
    where (@contrato = 'Sim' and CCNumber in (select CCNumber from NBA.CoachesWithContract))
        OR
        (@contrato = 'Nao' and CCNumber in (select CCNumber from NBA.CoachesWithoutContract))
        OR
        (@contrato is null)
);
go

-- Função com o filtro de conferencia das equipas
drop function IF EXISTS NBA.filtrarEquipasPorConferencia;
go
create function NBA.filtrarEquipasPorConferencia(@conferencia varchar(10)) returns table
as
return (
    select *
    from NBA.TeamCoachOwner
    where (Conference = @conferencia or @conferencia is null)
);
go

-- Função com o filtro de equipa da casa e equipa visitante
drop function IF EXISTS NBA.filtrarJogosPorEquipaCasaEquipaForaESeAconteceu
go
create function NBA.filtrarJogosPorEquipaCasaEquipaForaESeAconteceu(@equipaCasa varchar(30),
@equipaFora varchar(30), @aconteceu varchar(3)) returns table
as
return (
    select *
    from NBA.GamesTeamsStadium as G
```

```
    where (HomeTeamName = @equipaCasa or @equipaCasa is null) and (
            (@aconteceu = 'Sim' and G.ID in (select ID from NBA.GamesWithResult))
            OR
            (@aconteceu = 'Nao' and G.ID in (select ID from NBA.GamesWithoutResult))
            OR
            (@aconteceu is null)
        ) and (AwayTeamName = @equipaFora or @equipaFora is null)
);
go

-- Função que retorna a média de estatísticaa de um dado jogador
drop function IF EXISTS NBA.GetPlayerStats
go
create function NBA.GetPlayerStats(@playerCC int) returns table
as
return (
    select Points, Assists, Rebounds, Blocks, Steals, [FG%], [3PT%]
    from ((NBA.Average_Individual_Numbers as Stats inner join NBA.Player as Pl on Stats.Player_CCNumber
= Pl.CCNumber) inner join NBA.Person as Pe on Pl.CCNumber = Pe.CCNumber)
    WHERE Pe.CCNumber = @playerCC
);
go

-- Função que retorna a média de estatísticas de uma equipa
drop function IF EXISTS NBA.getTeamAverageStats
go
create function NBA.getTeamAverageStats(@InputTeamID INT) returns @TeamAverageStats table (
    TeamID int,
    TeamName varchar(50),
    AveragePoints decimal(10, 2),
    AverageAssists decimal(10, 2),
    AverageRebounds decimal(10, 2),
    AverageBlocks decimal(10, 2),
    AverageSteals decimal(10, 2),
    AverageFGP decimal(10, 2),
    Average3PTP decimal(10, 2)
)
as
    begin
        declare @TeamID as int;
        declare @TeamName varchar(50);
        declare @PlayersStats table
        (
            Points int,
            Assists int,
            Rebounds int,
            Blocks int,
            Steals int,
            [FG%] float,
            [3PT%] float
        );

        declare teamCursor cursor for select ID, [Name] from NBA.Team where ID = @InputTeamID;
        open teamCursor;

        fetch next from teamCursor into @TeamID, @TeamName;

        while @@FETCH_STATUS = 0
            begin
                insert into @PlayersStats (Points, Assists, Rebounds, Blocks, Steals, [FG%], [3PT%])
                    select Points, Assists, Rebounds, Blocks, Steals, [FG%], [3PT%]
                    from NBA.Average_Individual_Numbers as Stats inner join NBA.Player as Pl on
Stats.Player_CCNumber = Pl.CCNumber
                    where Pl.Team_ID = @TeamID;

                insert into @TeamAverageStats (TeamID, TeamName, AveragePoints, AverageAssists,
AverageRebounds, AverageBlocks, AverageSteals, AverageFGP, Average3PTP)
                    select @TeamID, @TeamName, avg(Points), avg(Assists), avg(Rebounds), avg(Blocks),
avg(Steals), avg([FG%]), avg([3PT%])
                    from @PlayersStats;

                delete from @PlayersStats;

                fetch next from teamCursor into @TeamID, @TeamName;
            end;

        close teamCursor;
        deallocate teamCursor;
```

```
        return;
    END;
go

-- Função para retornar a tabela de classificação
drop function IF EXISTS NBA.GetTeamStandings;
go
create function NBA.GetTeamStandings() returns @TeamStandings table (
    Team_ID int,
    Team_Name varchar(50),
        GamesPlayed int,
    Wins int,
    Losses int,
        [Win%] float
)
as
    begin
        -- Inserir os resultados dos jogos na tabela @GameWinners
        declare @GameWinners table (
            Game_ID int,
            Winner_ID int,
                        Loser_ID int
        );

        -- Chamar a UDF anterior para obter os vencedores de cada jogo
        insert into @GameWinners (Game_ID, Winner_ID, Loser_ID)
            select ID,
                (case
                    when Home_Score > Away_Score then Home_Team_ID
                    when Home_Score < Away_Score then Away_Team_ID
                end) as Winner_ID,
                                (case
                    when Home_Score > Away_Score then Away_Team_ID
                    when Home_Score < Away_Score then Home_Team_ID
                end) as Loser_ID
            from NBA.Game
                        where Home_Score is not null and Away_Score is not null;

        -- Calcular o número de vitórias e derrotas para cada equipe
                insert into @TeamStandings (Team_ID, Team_Name, GamesPlayed, Wins, Losses, [Win%])
                        select T.ID, T.[Name],
                                sum(case when GW.Winner_ID = T.ID then 1 else 0 end)+sum(case when
GW.Loser_ID = T.ID then 1 else 0 end) as GamesPlayed,
                                sum(case when GW.Winner_ID = T.ID then 1 else 0 end) as Wins,
                                sum(case when GW.Loser_ID = T.ID then 1 else 0 end) as Losses,
                                round((sum(case when GW.Winner_ID = T.ID then 1 else 0 end) * 100.0)
/ count(*) , 4) as [Win%]
                        from NBA.Team as T left join @GameWinners as GW on T.ID = GW.Winner_ID OR
T.ID = GW.Loser_ID
                        group by T.ID, T.[Name]
        return;
    end;
go

-- Função para retornar a tabela de jogos de uma dada equipa
drop function IF EXISTS NBA.GetTeamGames
go
create function NBA.GetTeamGames (@TeamID int) returns table
as
    return
    (
        select G.ID AS GameID, G.[Time], G.[Date], G.Home_Score, G.Away_Score, G.Home_Team_ID,
G.Away_Team_ID
        from (NBA.Game G inner join NBA.Team T on T.ID = G.Home_Team_ID or T.ID = G.Away_Team_ID)
        where T.ID = @TeamID
    )
go

-- Função para retornar os bilhetes de um dado jogo
drop function IF EXISTS NBA.GetGameTickets
go
create function NBA.GetGameTickets (@GameID int) returns table
as
return
(
    select [Type], Price, Restantes
    from NBA.Ticket
    where Game_ID = @GameID
)
```

Go

Stored Procedures

Os Stored Procedures são um recurso importante nas bases de dados que permitem armazenar e executar blocos de código SQL de forma organizada e reutilizável. Eles oferecem várias vantagens e utilidades, tais como uma performance aprimorada devido ao seu armazenamento em cache e à atomicidade quando se trata de transações complexas (ou todas as transações são executadas com sucesso ou nenhuma é).

Neste projeto, estas foram essencialmente usadas para o a manipulação dos dados da base de dados. Com isto, quer-se dizer que foram usados para adição, alteração ou exclusão de dados. Também foram usados na pesquisa pelo atributo nome em algumas tabelas.

Abaixo é apresentado o código dos diferentes Stored Procedures, tendo cada uma um comentário com a sua finalidade.

```
-- Procedure para barra de pesquisa
drop procedure IF EXISTS NBA.pesquisarPorNome;
go
create procedure NBA.pesquisarPorNome
            @nome varchar(50),
            @esquema varchar(50),
          @tabela varchar(50)
as
            begin
                        declare @query nvarchar(MAX);

                        set @query = 'select * from ' + QUOTENAME(@esquema) + '.' + QUOTENAME(@tabela) + ' where [Name] like ''%' + REPLACE(@nome, '''',
''''''') + '%''';

                        execute sp_executesql @query;
            end;
go

-- Procedure para adicionar ou alterar jogador
drop procedure IF EXISTS NBA.adicionarAlterarJogador;
go
create procedure NBA.adicionarAlterarJogador
    @CCNumber int,
    @Name varchar(50),
    @Age int,
    @Number int,
    @Height varchar(5),
    @Weight float,
    @Position varchar(20),
    @Team_ID int,
    @Contract_ID int = null,
    @Command varchar(20),
    @NumberOrTeamIDChanged varchar(3),
    @Points float = null,
    @Assists float = null,
    @Rebounds float = null,
    @Blocks float = null,
    @Steals float = null,
    @FG float = null,
    @PT3 float = null
as
            begin
                        declare @errorsCount as int = 0;

                        if (@NumberOrTeamIDChanged = 'Sim')
                                    if (@Number is not null and exists(select 1 from (NBA.Team as T inner join NBA.Player as P on T.ID = P.Team_ID)
where Team_ID = @Team_ID and Number = @Number))
                                                begin
                                                            set @errorsCount = @errorsCount + 1;
                                                            raiserror('Não foi possível adicionar/alterar jogador! Já existe um jogador da mesma
equipa com o mesmo número de equipamento.', 16, 1);
                                                end

                        if (@errorsCount = 0)
                                    begin
                                                if (@Command = 'adicionar')
                                                            begin try
                                                                        begin tran
                                                                                    insert into NBA.Person values(@CCNumber, @Name, @Age,
@Contract_ID);
                                                                                    insert into NBA.Player values(@CCNumber, @Number, @Height,
@Weight, @Position, @Team_ID);
                                                                                    insert into NBA.Average_Individual_Numbers values (@Points,
@Assists, @Rebounds, @Blocks, @Steals, @FG ,@PT3, @CCNumber);
                                                                        commit tran
                                                            end try
                                                            begin catch
                                                                        rollback tran
                                                                        raiserror('Jogador não inserido! Algum dado está incorreto', 16, 1);
                                                            end catch
                                                else if (@Command = 'alterar')
                                                            begin try
                                                                        begin tran
                                                                                    update NBA.Person
                                                                                    set [Name] = @Name, Age = @Age, Contract_ID = @Contract_ID
                                                                                    where CCNumber = @CCNumber;
```

```
                                                                                update NBA.Player
                                                                                set [Number] = @Number, Height = @Height, [Weight] = @Weight,
Position = @Position, Team_ID = @Team_ID
                                                                                where CCNumber = @CCNumber;
                                                                        commit tran
                                                                end try
                                                                begin catch
                                                                        rollback tran
                                                                        raiserror('Jogador não alterado! Algum dado está incorreto', 16, 1);
                                                                end catch
                                                                begin try
                                                                        begin tran
                                                                                update NBA.Average_Individual_Numbers
                                                                                set Points = @Points, Assists = @Assists, Rebounds = @Rebounds,
Blocks = @Blocks, Steals = @Steals, [FG%] = @FG, [3PT%] = @PT3
                                                                                where Player_CCNumber = @CCNumber;
                                                                        commit tran
                                                                end try
                                                                begin catch
                                                                        rollback tran
                                                                        raiserror('Estatística não alterads! Algum dado está incorreto', 16, 1);
                                                                end catch
                                        end
                end
go

-- Procedure para apagar jogador
drop procedure IF EXISTS NBA.apagarJogador;
go
create procedure NBA.apagarJogador
                @CCNumber int
as
                delete from NBA.Player where CCNumber = @CCNumber;
                delete from NBA.Person where CCNumber = @CCNumber;
go

-- Procedure para adicionar ou alterar treinador
drop procedure IF EXISTS NBA.adicionarAlterarTreinador;
go
create procedure NBA.adicionarAlterarTreinador
    @CCNumber int,
    @Name varchar(50),
    @Age int,
    @Contract_ID int = null,
                @Command varchar(20)
as
                begin
                        declare @errorsCount as int = 0;

                        if (@errorsCount = 0)
                                        begin
                                                        if (@Command = 'adicionar')
                                                                begin try
                                                                        begin tran
                                                                                insert into NBA.Person values(@CCNumber, @Name, @Age,
@Contract_ID);
                                                                                insert into NBA.Coach values(@CCNumber);
                                                                        commit tran
                                                                end try
                                                                begin catch
                                                                        rollback tran
                                                                        raiserror('Jogador não inserido! Algum dado está incorreto', 16, 1);
                                                                end catch
                                                        else if (@Command = 'alterar')
                                                                begin try
                                                                        begin tran
                                                                                update NBA.Person
                                                                                set [Name] = @Name, Age = @Age, Contract_ID = @Contract_ID
                                                                                where CCNumber = @CCNumber;
                                                                        commit tran
                                                                end try
                                                                begin catch
                                                                        rollback tran
                                                                        raiserror('Jogador não alterado! Algum dado está incorreto', 16, 1);
                                                                end catch
                                        end
                end
go

-- Procedure para apagar treinador
drop procedure IF EXISTS NBA.apagarTreinador;
go
create procedure NBA.apagarTreinador
                @CCNumber int
as
                delete from NBA.Coach where CCNumber = @CCNumber;
                delete from NBA.Person where CCNumber = @CCNumber;
go

-- Procedure para adicionar ou alterar equipa
drop procedure IF EXISTS NBA.adicionarAlterarEquipa;
go
create procedure NBA.adicionarAlterarEquipa
                @ID int = null,
    @Name varchar(50),
                @City varchar(50),
                @Conference varchar(50),
    @FoundYear int,
    @CoachCCNumber int,
                @OwnerCCNumber int,
                @Command varchar(20),
                @CoachChanged varchar(3)
as
                begin
                        declare @errorsCount as int = 0;
                        declare @nextID as int = (select max(ID)+1 from NBA.Team);
                        declare @nextIDStadium as int = (select max(ID)+1 from NBA.Stadium);
                        declare @s as varchar(7) = 'Stadium';
                        declare @nameStadium as varchar(50) = @Name+ ' ' + @s;
                        declare @nextIDContract as int = (select max(ID)+1 from NBA.[Contract]);
                        declare @c as varchar(8) = 'Contract';
                        declare @descriptionContract as varchar(50) = @Name+ ' ' + @c;

                        if ((@OwnerCCNumber is not null and exists (select 1 from NBA.Coach where CCNumber = @OwnerCCNumber)) or (@OwnerCCNumber is not
null and exists (select 1 from NBA.Player where CCNumber = @OwnerCCNumber)))
                                        begin
                                                        set @errorsCount = @errorsCount + 1;
                                                        raiserror('Não foi possível adicionar/alterar equipa! O presidente iserido é um treinaodr/jogador.',
16, 1);
```

```
                                        end

                                if (@CoachChanged = 'Sim')
                                        begin
                                                if (@CoachCCNumber is not null and exists (select 1 from NBA.Team where Coach_CCNumber =
@CoachCCNumber) or @CoachCCNumber is not null and exists (select 1 from NBA.Person where CCNumber = @CoachCCNumber and Contract_ID is not null))
                                                        begin
                                                                set @errorsCount = @errorsCount + 1;
                                                                raiserror('Não foi possível adicionar/alterar equipa! O treinador inserido já
pertence a outra equipa ou já tem contrato.', 16, 1);
                                                        end
                                        end

                                if (@errorsCount = 0)
                                        begin
                                                if (@Command = 'adicionar')
                                                        begin try
                                                                begin tran
                                                                        insert into NBA.Team values(@nextID, @Name, @City, @Conference,
@FoundYear, @OwnerCCNumber, @CoachCCNumber, 0);
                                                                        insert into NBA.Stadium values(@nextIDStadium, @nameStadium,
@City, 20000, @nextID);
                                                                        insert into NBA.[Contract] values(@nextIDContract,
@descriptionContract, 5000000, getdate(), dateadd(year, 5, getdate()));
                                                                        update NBA.Person set Contract_ID = @nextIDContract where
CCNumber = @CoachCCNumber;
                                                                commit tran
                                                        end try
                                                        begin catch
                                                                rollback tran
                                                                raiserror('Equipa não inserida! Algum dado está incorreto', 16, 1);
                                                        end catch
                                                else if (@Command = 'alterar')
                                                        begin try
                                                                begin tran
                                                                        update NBA.Team
                                                                        set [Name] = @Name, Conference = @Conference, Found_Year =
@FoundYear, Owner_CCNumber = @OwnerCCNumber, Coach_CCNumber = @CoachCCNumber
                                                                        where ID = @ID;
                                                                commit tran
                                                        end try
                                                        begin catch
                                                                rollback tran
                                                                raiserror('Equipa não alterado! Algum dado está incorreto', 16, 1);
                                                        end catch
                                        end
                end
go

-- Procedure para apagar equipa
drop procedure IF EXISTS NBA.apagarEquipa;
go
create procedure NBA.apagarEquipa
                @ID int
as
                begin
                        declare @coachCCNumber as int = (select Coach_CCNumber from NBA.Team where ID = @ID);

                        if exists(select * from NBA.Game where Home_Team_ID = @ID or Away_Team_ID = @ID)
                                begin
                                        begin try
                                                begin tran
                                                        update NBA.Team set disabled = 1 where ID = @ID;
                                                        update NBA.Person set Contract_ID = null where CCNumber = @coachCCNumber;
                                                commit tran
                                        end try
                                        begin catch
                                                rollback tran
                                                raiserror('Erro! Equipa não desativada', 16, 1);
                                        end catch

                                end
                        else
                                begin
                                        begin try
                                                begin tran
                                                        delete from NBA.Team where ID = @ID;
                                                        update NBA.Person set Contract_ID = null where CCNumber = @coachCCNumber;
                                                commit tran
                                        end try
                                        begin catch
                                                rollback tran
                                                raiserror('Erro! Equipa não apagada', 16, 1);
                                        end catch
                                end
                end
go

-- Procedure para adicionar ou alterar jogo
drop procedure IF EXISTS NBA.adicionarAlterarJogo;
go
create procedure NBA.adicionarAlterarJogo
                @ID int = null,
        @Time time,
                @Date date,
                @HomeScore int = null,
        @AwayScore int = null,
        @HomeTeamID int,
                @AwayTeamID int,
                @StadiumID int,
                @Command varchar(30)
as
                begin
                        declare @errorsCount as int = 0;
                        declare @nextID as int = (select max(ID)+1 from NBA.Game);

                        if ((@HomeScore is null and @AwayScore is not null) or (@HomeScore is not null and @AwayScore is null))
                                begin
                                        set @errorsCount = @errorsCount + 1;
                                        raiserror('Não foi possível adicionar/alterar jogo! O resultado está incompleto.', 16, 1);
                                end

                        if (@HomeTeamID = @AwayTeamID)
                                begin
                                        set @errorsCount = @errorsCount + 1;
                                        raiserror('Não foi possível adicionar/alterar jogo! O jogo tem de ser entre equipas diferentes.', 16,
1);
                                end
```

```
                        if (@HomeTeamID != @StadiumID and @AwayTeamID != @StadiumID)
                                    begin
                                                set @errorsCount = @errorsCount + 1;
                                                raiserror('Não foi possível adicionar/alterar jogo! A arena tem de pertencer a uma das equipas.', 16,
1);
                                    end

                        if (@Date < getdate() and @HomeScore is null and @AwayScore is null)
                                    begin
                                                set @errorsCount = @errorsCount + 1;
                                                raiserror('Não foi possível adicionar jogo! Como o jogo já aconteceu tem de haver resultado.', 16, 1);
                                    end

                        if (@errorsCount = 0)
                                    begin
                                                if (@Command = 'adicionar')
                                                            begin try
                                                                        begin tran
                                                                                    insert into NBA.Game values(@nextID, @Time, @Date, @HomeScore,
@AwayScore, @HomeTeamID, @AwayTeamID, @StadiumID);
                                                                        commit tran
                                                            end try
                                                            begin catch
                                                                        rollback tran
                                                                        raiserror('Jogo não inserido! Algum dado está incorreto', 16, 1);
                                                            end catch
                                                else if (@Command = 'alterar')
                                                            begin try
                                                                        begin tran
                                                                                    update NBA.Game
                                                                                    set [Time]  = @Time, [Date] = @Date, Home_Score = @HomeScore,
Away_Score = @AwayScore, Home_Team_ID = @HomeTeamID,  Away_Team_ID = @AwayTeamID, Stadium_ID = @StadiumID
                                                                                    where ID = @ID;
                                                                        commit tran
                                                            end try
                                                            begin catch
                                                                        rollback tran
                                                                        raiserror('Jogo não alterado! Algum dado está incorreto', 16, 1);
                                                            end catch
                                    end
            end
go

-- Procedure para apagar jogo
drop procedure IF EXISTS NBA.apagarJogo;
go
create procedure NBA.apagarJogo
            @ID int
as
            delete from NBA.Ticket where Game_ID = @ID;
            delete from NBA.Game where ID = @ID;
go

-- Procedure para adicionar ou alterar bilhetes de jogos
drop procedure IF EXISTS NBA.adicionarAlterarBilhetes;
go
create procedure NBA.adicionarAlterarBilhetes
            @Type varchar(30),
    @Price float,
            @Restantes int,
            @Game_ID int,
            @Team_ID int,
            @Command varchar(30)
as
            begin
                        declare @errorsCount as int = 0;

                        if (@errorsCount = 0)
                                    begin
                                                if (@Command = 'adicionar')
                                                            begin try
                                                                        begin tran
                                                                                    insert into NBA.Ticket values(@Type, @Price, @Restantes,
@Game_ID, @Team_ID);
                                                                        commit tran
                                                            end try
                                                            begin catch
                                                                        rollback tran
                                                                        raiserror('Bilhetes não inseridos! Algum dado está incorreto', 16, 1);
                                                            end catch
                                                else if (@Command = 'alterar')
                                                            begin try
                                                                        begin tran
                                                                                    update NBA.Ticket
                                                                                    set Price = @Price, Restantes = @Restantes
                                                                                    where [Type] = @Type and Game_ID = @Game_ID;
                                                                        commit tran
                                                            end try
                                                            begin catch
                                                                        rollback tran
                                                                        raiserror('Bilhetes não alterados! Algum dado está incorreto', 16, 1);
                                                            end catch
                                    end
            end
go
```

Triggers

Os Triggers são objetos de base de dados que são acionados automaticamente em resposta a determinados eventos, como inserção, atualização ou exclusão de dados numa tabela. Eles oferecem várias vantagens e utilidades que auxiliam no controle, automação e manutenção da integridade dos dados.

No nosso projeto, não sentimos a necessidade de utilização destes, pois maior parte das verificações já estão efetuadas na DDL e as verificações mais complexas foram efetuadas nas Stored Procedures.

Interface

Apesar de não ter sido um ponto fulcral do projeto, foi criada uma interface mais intuitiva possível e que proporcionasse várias funcionalidades para a demonstração de todo o trabalho desenvolvido.

Apenas foi criado o formulário para o administrador, pois é neste onde todas as funcionalidades podem ser vistas, sendo que o formulário do cliente seria parecido, não permitindo a alteração dos dados da base de dados, servindo assim só de visualização.

Neste formulário exstem 4 tabs, um dos jogadores, outro dos treinadores, outro das equipas e outro dos jogos.

No tab dos jogadores, é possível visualizar uma lista com todos os jogadores que, clicando em algum, são apresentados os seus dados. Também existem diversos filtros, como a pesquisa por nome, filtro por equipa, existência de contrato e posição de jogo. É possível a adição, alteração e exclusão de jogadores.

No tab dos treinadores, é possível visualizar uma lista com todos os treinadores que, clicando em algum, são apresentados os seus dados. Existem filtros como a pesquisa por nome e existência de contrato. É possível a adição, alteração e exclusão de treinadores.

No tab das equipas, é possível visualizar uma lista com todas as equipas que, clicando em alguma, são apresentados os seus dados. Adicionalmente, também são apresentados os jogos dessa equipa e a sua estatística média. Existem filtros como a pesquisa por nome e a conferência a que pertence. É possível a adição, alteração e exclusão equipas.

No tab dos jogos, é possível visualizar uma lista com todas os jogos do campeonato que, clicando em algum, são apresentados os seus dados. Para além dos jogos, também a apresentada a tabela classificativa. Existem filtros como a equipa que joga em casa, a equipa que joga fora e se o jogo já aconteceu. É possível a adição, alteração e exclusão jogos.

É de notar que todos estes processos de adições, alterações e remoções estão corretamente sincronizados para a interface atualizar logo e serem evidentes as mudanças.

Conclusão

Com este projeto foi possível verificar na prática o funcionamento e a utilidade de um Sistema de Gestão de Base de Dados (SGBD). O projeto permitiu compreender a importância de ter um local centralizado para armazenar e gerenciar informações, garantindo a consistência e a segurança dos dados.

Embora o projeto não tenha focado na interface gráfica, a existência de uma interface intuitiva facilitou a visualização e a interação com os dados, proporcionando uma experiência mais fluída para os usuários.

Em resumo, os objetivos propostos inicialmente foram maioritariamente alcançados, com ajustes e adições de recursos ao longo do projeto. Esses esforços permitiram aprofundar o conhecimento em relação aos conceitos e práticas abordados na disciplina de Base de Dados.

Portanto, o projeto serviu como uma oportunidade de consolidar o aprendizado teórico por meio da aplicação prática, além de evidenciar a importância de um SGBD para o gerenciamento eficiente e seguro das informações.