Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Аналіз даних в інформаційно-управляючих системах

ЗВІТ

до лабораторної роботи №1

Виконав
студент **ІП-01 Хернуф Валід Алі-Еддін**
(№ групи, прізвище, ім'я, по батькові)

Київ 2022

1. Спроектувати модель Stage зони для ETL процесів

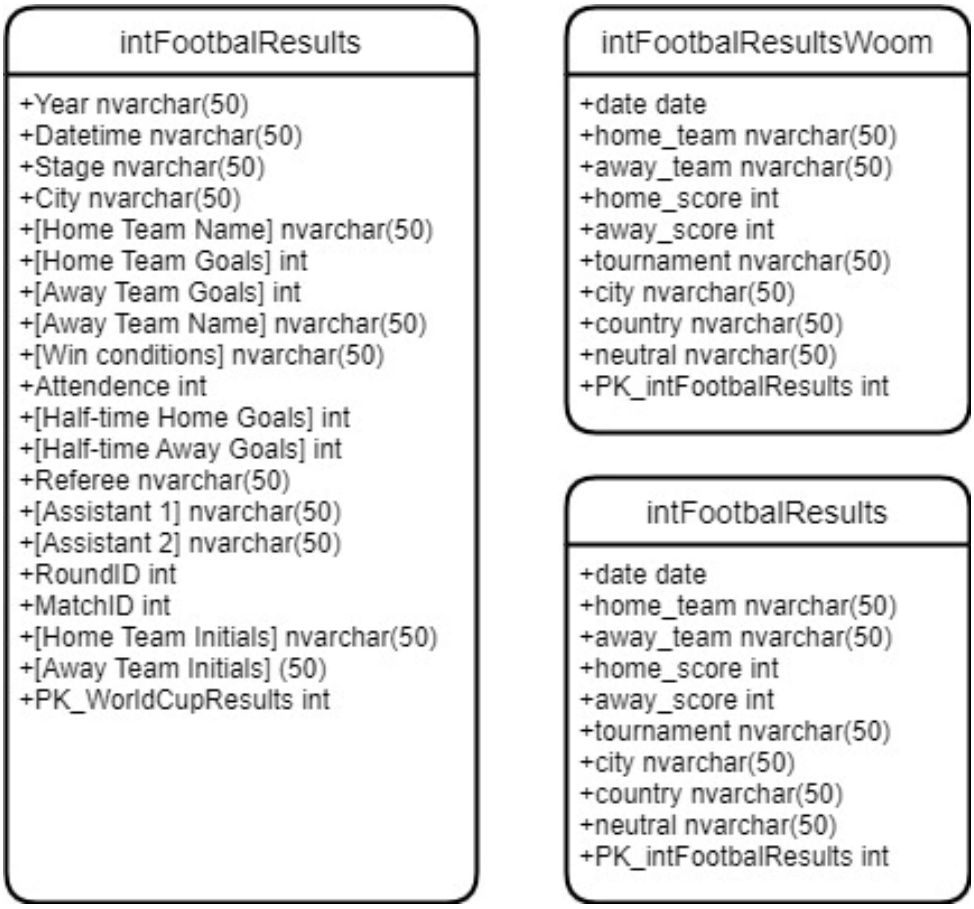

Рис.1 – ER stage zone

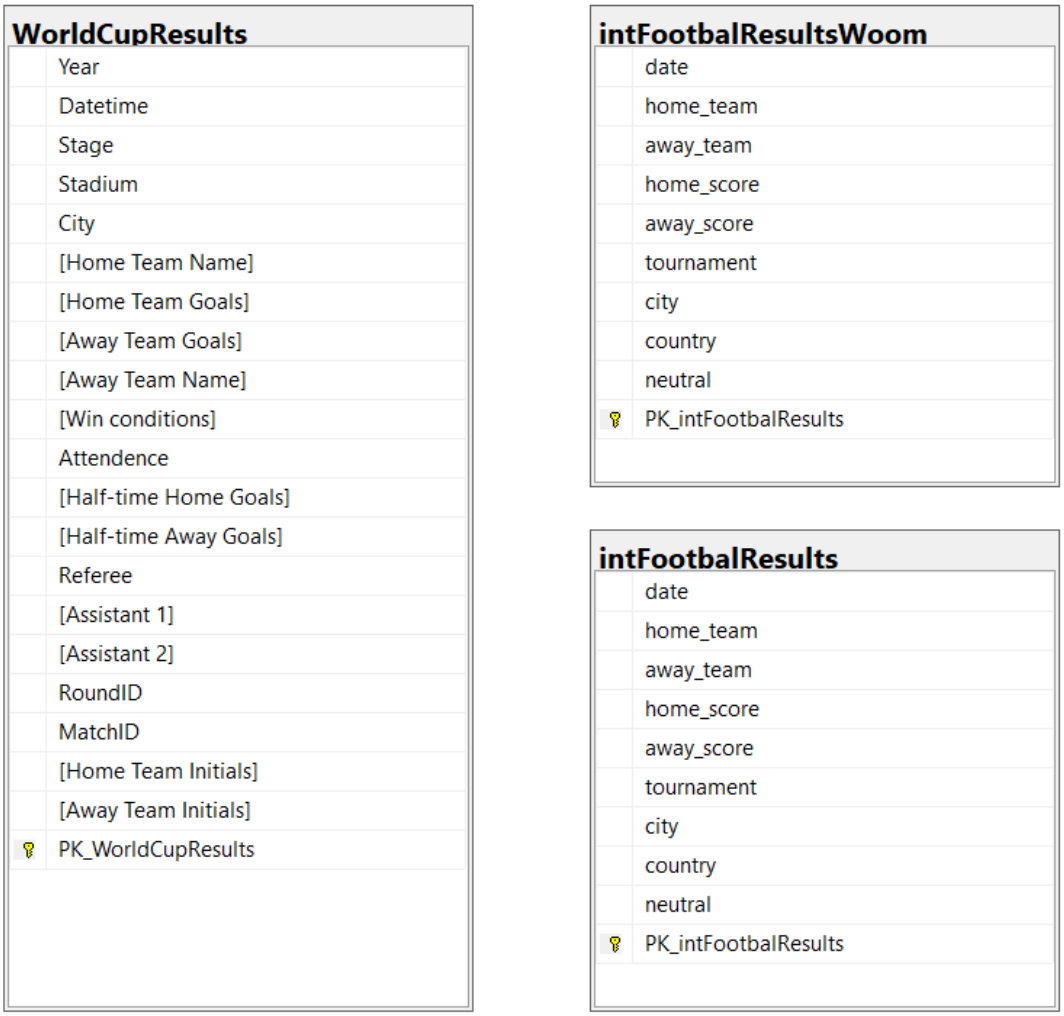

Рис 2. – Stage zone scheme from MS SQL

Скрипти створення Stage зони знаходяться у додатку А.

Опис таблиць stage зони:

1. WorldCupResults – таблиця містить інформацію про матчі з чемпіонатів світу: дату, стадію, стадіон, місто, назву першої команди, назву другої команди, кількість голів двох команд, умову перемоги, ім’я рефері та асистентів.
2. intFootbalResults – таблиця містить інформацію про звичайні міжнародні матчі: дату, назви команд, кількість голів, назву турніру, місто, країну, та інформацію чи був цей матч на нейтральній території.
3. intFootbalResultsWoom – таблиця містить інформацію про жіночі звичайні міжнародні матчі: дату, назви команд, кількість голів, назву турніру, місто, країну, та інформацію чи був цей матч на нейтральній території.

2. **Спроектувати модель основного сховища за типом за типом «зірка» або «сніжинка».**

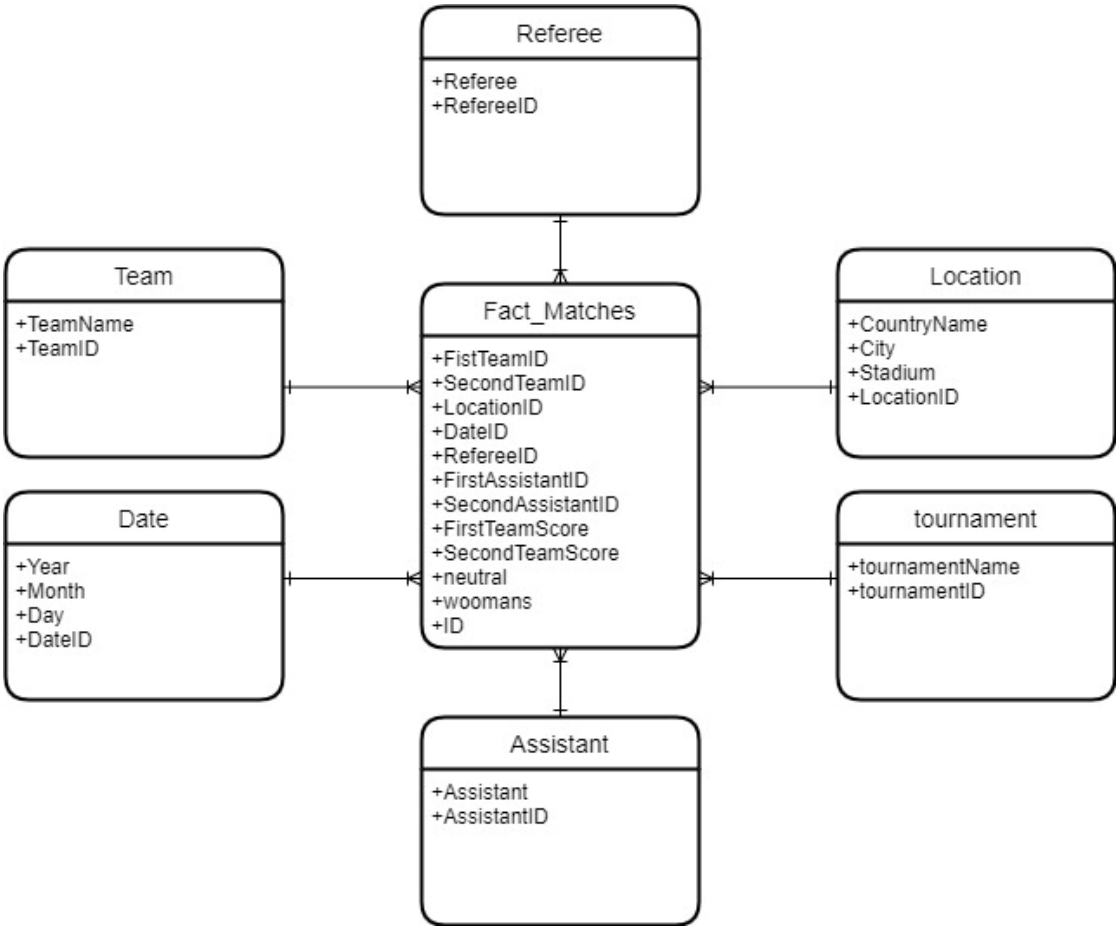

Рис.3 – ER Star database

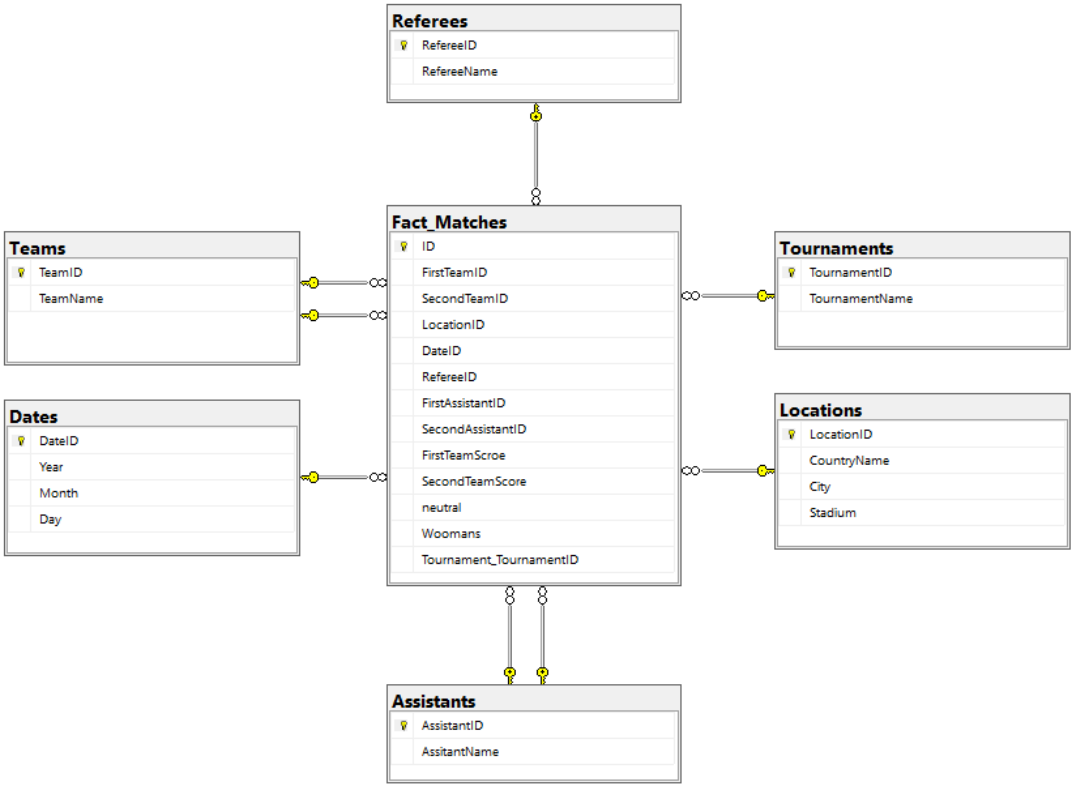

Рис.3 – Star database scheme from MS SQL

Скрипти створення основного сховища за типом «зірка» знаходяться у додатку Б.

Опис таблиць сховища:

1. Teams - вимір “Команди”, що містить інформацію про футбольні команди країн.
2. Dates – вимір “Дати”, що містить інформацію про дату (рік, місяць та день) матчів.
3. Assistants – вимір “Асистенти”, що містить інформацію (ім’я та прізвище) про асистентів рефері.
4. Referee – вимір “Рефері”, що містить інформацію (ім’я та прізвище) про рефері.
5. Locations – вимір “Локації”, що містить інформацію (країну, місто та стадіон) про місце проведення матчу.
6. Tournament – вимір “Турнір”, що містить інформацію про турніри.
7. Fact_Matches – фактова таблиця для збереження фактів про матчі, такі як: кількість голів першої команди, кількість голів другої команди, нейтральність та чи був матч жіночім.

3. Створити ETL засоби

Скрипти ETL-процедур знаходяться у додатку В.

Додаток А

(Скрипти для створення Stage зони)

```
/*=======================================*
*            Table: IntFootbalResultsWoom
*=======================================*/

CREATE TABLE [dbo].[intFootbalResultsWoom](
       [date] [date] NOT NULL,
       [home_team] [nvarchar](50) NOT NULL,
       [away_team] [nvarchar](50) NOT NULL,
       [home_score] [int] NOT NULL,
       [away_score] [int] NOT NULL,
       [tournament] [nvarchar](50) NOT NULL,
       [city] [nvarchar](50) NOT NULL,
       [country] [nvarchar](50) NOT NULL,
       [neutral] [nvarchar](50) NOT NULL,
       [PK_intFootbalResults] int IDENTITY(1,1) NOT NULL PRIMARY KEY CLUSTERED
) ON [PRIMARY]
GO

/*=======================================*
*            Table: IntFootbalResults
*=======================================*/

CREATE TABLE [dbo].[intFootbalResults](
       [date] [date] NOT NULL,
       [home_team] [nvarchar](50) NOT NULL,
       [away_team] [nvarchar](50) NOT NULL,
       [home_score] [int] NOT NULL,
       [away_score] [int] NOT NULL,
       [tournament] [nvarchar](50) NOT NULL,
       [city] [nvarchar](50) NOT NULL,
       [country] [nvarchar](50) NOT NULL,
       [neutral] [nvarchar](50) NOT NULL,
       [PK_intFootbalResults] int IDENTITY(1,1) NOT NULL PRIMARY KEY CLUSTERED
) ON [PRIMARY]
GO


/*=======================================*
*            Table: WorldCupResults
*=======================================*/

CREATE TABLE [dbo].[WorldCupResults](
       [Year] [nvarchar](50) NOT NULL,
       [Datetime] [nvarchar](50) NOT NULL,
       [Stage] [nvarchar](50) NOT NULL,
       [Stadium] [nvarchar](50) NOT NULL,
       [City] [nvarchar](50) NOT NULL,
       [Home Team Name] [nvarchar](50) NOT NULL,
       [Home Team Goals] [int] NOT NULL,
       [Away Team Goals] [int] NOT NULL,
       [Away Team Name] [nvarchar](50) NOT NULL,
       [Win conditions] [nvarchar](50) NULL,
       [Attendence] [int] NULL,
       [Half-time Home Goals] [int] NOT NULL,
       [Half-time Away Goals] [int] NOT NULL,
       [Referee] [nvarchar](50) NOT NULL,
       [Assistant 1] [nvarchar](50) NOT NULL,
       [Assistant 2] [nvarchar](50) NOT NULL,
       [RoundID] [int] NOT NULL,
       [MatchID] [int] NOT NULL,
```

```
	[Home Team Initials] [nvarchar](50) NOT NULL,
	[Away Team Initials] [nvarchar](50) NOT NULL,
	[PK_WorldCupResults] int IDENTITY(1,1) NOT NULL PRIMARY KEY CLUSTERED
) ON [PRIMARY]
GO

/*=======================================*
*            Uploading data to StageZones
*=======================================*/

CREATE TABLE #csv_temp(
	[date] [date] NOT NULL,
	[home_team] [nvarchar](50) NOT NULL,
	[away_team] [nvarchar](50) NOT NULL,
	[home_score] [int] NOT NULL,
	[away_score] [int] NOT NULL,
	[tournament] [nvarchar](50) NOT NULL,
	[city] [nvarchar](50) NOT NULL,
	[country] [nvarchar](50) NOT NULL,
	[neutral] [nvarchar](50) NOT NULL,
) ON [PRIMARY]
GO

BULK INSERT #csv_temp
FROM 'C:\Users\admin\Desktop\KPI\FourthSemestr\ADIS\1lab\datasets\sets\results.csv'
WITH (
	FIRSTROW = 2,
	FIELDTERMINATOR = ',',
	ROWTERMINATOR = '0x0a'
	);
GO

INSERT INTO [dbo].[intFootbalResults] ([date], home_team, away_team, home_score,
away_score, tournament, city, country, neutral)
SELECT * FROM #csv_temp
GO

CREATE TABLE #csv_temp_woom(
	[date] [date] NOT NULL,
	[home_team] [nvarchar](50) NOT NULL,
	[away_team] [nvarchar](50) NOT NULL,
	[home_score] [int] NOT NULL,
	[away_score] [int] NOT NULL,
	[tournament] [nvarchar](50) NOT NULL,
	[city] [nvarchar](50) NOT NULL,
	[country] [nvarchar](50) NOT NULL,
	[neutral] [nvarchar](50) NOT NULL,
) ON [PRIMARY]
GO

BULK INSERT #csv_temp_woom
FROM 'C:\Users\admin\Desktop\KPI\FourthSemestr\ADIS\1lab\datasets\sets\woomresults.csv'
WITH (
	FIRSTROW = 2,
	FIELDTERMINATOR = ',',
	ROWTERMINATOR = '0x0a'
	);
GO

INSERT INTO [dbo].[intFootbalResultsWoom] ([date], home_team, away_team, home_score,
away_score, tournament, city, country, neutral)
SELECT * FROM #csv_temp_woom
GO

CREATE TABLE #csv_temp_world(
```

```
	[Year] [nvarchar](50) NOT NULL,
	[Datetime] [nvarchar](50) NOT NULL,
	[Stage] [nvarchar](50) NOT NULL,
	[Stadium] [nvarchar](50) NOT NULL,
	[City] [nvarchar](50) NOT NULL,
	[Home Team Name] [nvarchar](50) NOT NULL,
	[Home Team Goals] [int] NOT NULL,
	[Away Team Goals] [int] NOT NULL,
	[Away Team Name] [nvarchar](50) NOT NULL,
	[Win conditions] [nvarchar](50) NULL,
	[Attendence] [int] NULL,
	[Half-time Home Goals] [int] NOT NULL,
	[Half-time Away Goals] [int] NOT NULL,
	[Referee] [nvarchar](50) NOT NULL,
	[Assistant 1] [nvarchar](50) NOT NULL,
	[Assistant 2] [nvarchar](50) NOT NULL,
	[RoundID] [int] NOT NULL,
	[MatchID] [int] NOT NULL,
	[Home Team Initials] [nvarchar](50) NOT NULL,
	[Away Team Initials] [nvarchar](50) NOT NULL,
) ON [PRIMARY]
GO

BULK INSERT #csv_temp_world
FROM
'C:\Users\admin\Desktop\KPI\FourthSemestr\ADIS\1lab\datasets\sets\WorldCupMatches.csv'
WITH (
	FIRSTROW = 2,
	FIELDTERMINATOR = ';',
	ROWTERMINATOR = '0x0a'
	);
GO

INSERT INTO [dbo].[WorldCupResults] (
	[Year],
	[Datetime],
	Stage,
	Stadium,
	City,
	[Home Team Name],
	[Home Team Goals],
	[Away Team Goals],
	[Away Team Name],
	[Win conditions],
	Attendence,
	[Half-time Home Goals],
	[Half-time Away Goals],
	Referee,
	[Assistant 1],
	[Assistant 2],
	RoundID,
	MatchID,
	[Home Team Initials],
	[Away Team Initials]
	)
SELECT * FROM #csv_temp_world
GO
```

Додаток Б

(Створення сховища даних за типом «Зірка» за допомогою C#)

```
/*================================

*      Сховище даних в цілому

*================================*/



namespace Adis_FirstLab
{
    public class footbal_StarEntities : DbContext
    {
        public footbal_StarEntities()
            : base("name=BloggingCompactDatabase")
        {
        }

        public virtual DbSet<intFootbalResults> intFootbalResults { get; set; }
        public virtual DbSet<intFootbalResultsWoom> intFootbalResultsWoom { get; set; }
        public virtual DbSet<WorldCupResults> WorldCupResults { get; set; }
        public virtual DbSet<Team> Team { get; set; }
        public virtual DbSet<Referee> Referee { get; set; }
        public virtual DbSet<Assistant> Assistant { get; set; }
        public virtual DbSet<Tournament> Tournament { get; set; }
        public virtual DbSet<Date> Date { get; set; }
        public virtual DbSet<Location> Location { get; set; }
        public virtual DbSet<Fact_Matches> Fact_Matches { get; set; }
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Fact_Matches>()
                        .HasRequired(m => m.FirstTeam)
                        .WithMany(t => t.FirstTeams)
                        .HasForeignKey(m => m.FirstTeamID)
                        .WillCascadeOnDelete(false);
            modelBuilder.Entity<Fact_Matches>()
                        .HasRequired(m => m.SecondTeam)
                        .WithMany(t => t.SecondTeams)
                        .HasForeignKey(m => m.SecondTeamID)
                        .WillCascadeOnDelete(false);
            modelBuilder.Entity<Fact_Matches>()
                        .HasRequired(m => m.FirstAssistnant)
                        .WithMany(t => t.FirstAssistants)
                        .HasForeignKey(m => m.FirstAssistantID)
                        .WillCascadeOnDelete(false);
            modelBuilder.Entity<Fact_Matches>()
                        .HasRequired(m => m.SecondAssistant)
                        .WithMany(t => t.SecondAssistants)
                        .HasForeignKey(m => m.SecondAssistantID)
                        .WillCascadeOnDelete(false);
        }
    }
}
```

```
/*=========================

*      Table: Fact_Matches

*=========================*/

namespace Adis_FirstLab
{
    public class Fact_Matches
    {
        public int? FirstTeamID { get; set; }
        public int? SecondTeamID { get; set; }
        public int? LocationID { get; set; }
        public int? DateID { get; set; }
        public int? RefereeID { get; set; }
        public int? FirstAssistantID { get; set; }
        public int? SecondAssistantID { get; set; }
        [Required]
        public int FirstTeamScroe { get; set; }
        [Required]
        public int SecondTeamScore { get; set; }
        public bool neutral { get; set; }
        [Required]
        public bool Woomans { get; set; }
        [Key]
        public int ID { get; set; }

        //Navigation property

        public virtual Team FirstTeam { get; set; }
        public virtual Team SecondTeam { get; set; }
        [ForeignKey("LocationID")]
        public virtual Location Location { get; set; }
        [ForeignKey("DateID")]
        public virtual Date Date { get; set; }
        [ForeignKey("RefereeID")]
        public virtual Referee Referee { get; set; }
        public virtual Assistant FirstAssistnant { get; set; }
        public virtual Assistant SecondAssistant { get; set; }
    }
}

/*=========================

*      Table: Date

*=========================*/

namespace Adis_FirstLab
{
    public class Date
    {
        [Required]
        public int Year { get; set; }
        [Required]
        public int Month { get; set; }
        [Required]
        public int Day { get; set; }
        [Key]
        public int DateID { get; set; }

        //Navigation property

        public ICollection<Fact_Matches> Match_Dates { get; set; }
    }
}
```

```
/*=========================

*      Table: Assistant

*=========================*/

namespace Adis_FirstLab
{
    public class Assistant
    {
        [Required]
        [MaxLength(50)]
        public string AssitantName { get; set; }
        [Key]
        public int AssistantID { get; set; }

        //Navigation property

        public ICollection<Fact_Matches> FirstAssistants { get; set; }
        public ICollection<Fact_Matches> SecondAssistants { get; set; }
    }
}

/*=========================

*      Table: Location

*=========================*/

namespace Adis_FirstLab
{
    public class Location
    {
        [Required]
        [MaxLength(50)]
        public string CountryName { get; set; }
        [Required]
        [MaxLength(50)]
        public string City { get; set; }
        [MaxLength(50)]
        public string Stadium { get; set; }
        [Key]
        public int LocationID { get; set; }

        //Navigation property

        public ICollection<Fact_Matches> Match_Locations { get; set; }
    }
}

/*=========================

*      Table: Referee

*=========================*/

namespace Adis_FirstLab
{
    public class Referee
    {
        [Required]
        [MaxLength(50)]
        public string RefereeName { get; set; }
        [Key]
        public int RefereeID { get; set; }

        //Navigation property
```

```
        public ICollection<Fact_Matches> Match_Referees { get; set; }
    }
}

/*=========================

*      Table: Team

*=========================*/

namespace Adis_FirstLab
{
    public class Team
    {
        [Required]
        [MaxLength(50)]
        public string TeamName { get; set; }
        [Key]
        public int TeamID { get; set; }

        //Navigation property

        public ICollection<Fact_Matches> FirstTeams { get; set; }
        public ICollection<Fact_Matches> SecondTeams { get; set; }
    }
}

/*=========================

*      Table: Tournament

*=========================*/

namespace Adis_FirstLab
{
    public class Tournament
    {
        [Required]
        [MaxLength(50)]
        public string TournamentName { get; set; }
        [Key]
        public int TournamentID { get; set; }

        //Navigation property

        public ICollection<Fact_Matches> Match_Tournament { get; set; }
    }
}
```

Додаток В

(ETL скрипти)

```
namespace Adis_FirstLab
{
    public class Program
    {
        static void Main(string[] args)
        {
            using (var context = new footbal_StarEntities())
            {
                ETL.FillAssistantTable(context);
                ETL.FillDateTable(context);
                ETL.FillLocationTable(context);
                ETL.FillRefereeTable(context);
                ETL.FillTeamTable(context);
                ETL.FillTournamentTable(context);
                ETL.FillFactTable(context);
                context.SaveChanges();
            }
        }
    }
}


namespace Adis_FirstLab
{
    static class ETL
    {
        //Filling Date Table
        public static void FillDateTable(footbal_StarEntities context)
        {
            var allDatesInt = context.intFootbalResults.Select(d => d.date)

.Union(context.intFootbalResultsWoom
                                                                    .Select(d =>
d.date));
            var worldCupDates = context.WorldCupResults.Select(d => d.Datetime)
                                                       .Distinct()
                                                       .AsEnumerable()
                                                       .Select(d => new DateTime
(StrToDateTime(d).Item1, StrToDateTime(d).Item2, StrToDateTime(d).Item3));
            var allDates = allDatesInt.Union(worldCupDates).ToList();
            allDates.Sort();
            foreach(var date in allDates)
            {
                Date current = new Date() { Year = date.Year, Month = date.Month, Day =
date.Day };
                context.Date.Add(current);
            }
        }

        //Filling Referee Table
        public static void FillRefereeTable(footbal_StarEntities context)
        {
            var allReferies = from matches in context.WorldCupResults
                              select matches.Referee;
            allReferies = allReferies.AsQueryable().Distinct();
            foreach (string referee in allReferies)
            {
                Referee current = new Referee() { RefereeName = referee };
                context.Referee.Add(current);
```

```
            }
        }

        //Filling Assistant Table
        public static void FillAssistantTable(footbal_StarEntities context)
        {
            var allAssistants = context.WorldCupResults.Select(a => a.Assistant_1)
                                                       .Union(context.WorldCupResults
                                                                     .Select(a =>
a.Assistant_2));
            foreach (var assist in allAssistants)
            {
                Assistant current = new Assistant() { AssitantName = assist };
                context.Assistant.Add(current);
            }
        }

        //Filling Tournament Table
        public static void FillTournamentTable(footbal_StarEntities context)
        {
            var allTournaments = context.intFootbalResults.Select(t => t.tournament)

.Union(context.intFootbalResultsWoom
                                                                           .Select(t =>
t.tournament))
                                                          .AsQueryable()
                                                          .Distinct();
            foreach (var tourn in allTournaments)
            {
                Tournament current = new Tournament() { TournamentName = tourn };
                context.Tournament.Add(current);
            }
        }

        //Filling Team Table
        public static void FillTeamTable(footbal_StarEntities context)
        {
            var allTeamsMens = context.intFootbalResults
                                      .Select(t => t.home_team)
                                      .Union(context.intFootbalResults
                                                    .Select(t => t.away_team));
            var allTeamsWooms = context.intFootbalResultsWoom
                                      .Select(t => t.home_team)
                                      .Union(context.intFootbalResultsWoom
                                                    .Select(t => t.away_team));
            var allTeamsWorldCup = context.WorldCupResults
                                      .Select(t => t.Home_Team_Name)
                                      .Union(context.WorldCupResults
                                                    .Select(t => t.Away_Team_Name));
            var allTeams =
allTeamsMens.Union(allTeamsWooms).Union(allTeamsWorldCup).ToList();
            for (int i = 0; i < allTeams.Count; i++)
            {
                allTeams[i] = ClearName(allTeams[i]);
            }
            allTeams = allTeams.AsQueryable().Distinct().ToList();
            foreach (var team in allTeams)
            {
                Team current = new Team() { TeamName = team };
                context.Team.Add(current);
            }
        }

        //Filling Location Table
        public static void FillLocationTable(footbal_StarEntities context)
        {
```

```
            var locationsMen = context.intFootbalResults.Select(l => new { country =
l.country, city = l.city })
                                                        .Distinct()
                                                        .AsEnumerable()
                                                        .Select(l => new Loc { Country =
l.country, City = l.city });
            var locationsWoom = context.intFootbalResultsWoom.Select(l => new { country =
l.country, city = l.city })
                                                         .Distinct()
                                                         .AsEnumerable()
                                                         .Select(l => new Loc { Country =
l.country, City = l.city });
            var locationsWorld = context.WorldCupResults.Select(l => new { year = l.Year,
city = l.City, stadium = l.Stadium })
                                                        .Distinct()
                                                        .AsEnumerable()
                                                        .Select(l => new Loc { Country =
getWorldCupHost(Convert.ToInt32(l.year), l.city), City = l.city, Stadium = l.stadium });
            var allLocations = locationsMen.Union(locationsWoom).Union(locationsWorld);
            foreach (var location in allLocations)
            {
                Location current = new Location() { CountryName = location.Country, City
= location.City, Stadium = location.Stadium };
                context.Location.Add(current);
            }
        }

        //Filling Fact Table
        public static void FillFactTable(footbal_StarEntities context)
        {
            var matchMen = context.intFootbalResults.Select(m => m)
                                                    .AsEnumerable()
                                                    .Select(m => new Fact
                                                    {
                                                        Date = m.date,
                                                        HomeTeam =
ClearName(m.home_team),
                                                        AwayTeam =
ClearName(m.away_team),
                                                        HomeScore = m.home_score,
                                                        AwayScore = m.away_score,
                                                        Tourn = m.tournament,
                                                        City = m.city,
                                                        Country = m.country,
                                                        Wooms = false
                                                    });
            var matchWooms = context.intFootbalResultsWoom.Select(m => m)
                                                      .AsEnumerable()
                                                      .Select(m => new Fact
                                                      {
                                                          Date = m.date,
                                                          HomeTeam =
ClearName(m.home_team),
                                                          AwayTeam =
ClearName(m.away_team),
                                                          HomeScore = m.home_score,
                                                          AwayScore = m.away_score,
                                                          Tourn = m.tournament,
                                                          City = m.city,
                                                          Country = m.country,
                                                          Wooms = true
                                                      });
            var matchWorldCup = context.WorldCupResults.Select(m => m)
                                                    .AsEnumerable()
                                                    .Select(m => new Fact
```

```
                                                    {
                                                        Date = new
DateTime(StrToDateTime(m.Datetime).Item1, StrToDateTime(m.Datetime).Item2,
StrToDateTime(m.Datetime).Item3),
                                                        HomeTeam =
ClearName(m.Home_Team_Name),
                                                        AwayTeam =
ClearName(m.Away_Team_Name),
                                                        HomeScore = m.Home_Team_Goals,
                                                        AwayScore = m.Away_Team_Goals,
                                                        Tourn = "FIFA World Cup",
                                                        City = m.City,
                                                        Country =
getWorldCupHost(Convert.ToInt32(m.Year), m.City),
                                                        Stadium = m.Stadium,
                                                        Wooms = false,
                                                        Assistant1 = m.Assistant_1,
                                                        Assistant2 = m.Assistant_2,
                                                        Referee = m.Referee
                                                    });
            var allMatches =
matchMen.Union(matchWooms).Union(matchWorldCup).Distinct().ToList();
            List<Fact_Matches> matchs = new List<Fact_Matches>();
            int i = 0;
            foreach (var match in allMatches)
            {
                i++;
                match.FindIDs(context);
                Fact_Matches current = new Fact_Matches {
                                                         FirstTeamID =
match.HomeTeamID,
                                                         SecondTeamID =
match.AwayTeamID,
                                                         LocationID = match.LocID,
                                                         DateID = match.DateID,
                                                         RefereeID = match.RefereeID,
                                                         TournamentID = match.TournID,
                                                         FirstAssistantID =
match.Assistant1ID,
                                                         SecondAssistantID =
match.Assistant2ID,
                                                         FirstTeamScore =
match.HomeScore,
                                                         SecondTeamScore =
match.AwayScore,
                                                         neutral = match.neutral,
                                                         Woomans = match.Wooms
                                                     };
                matchs.Add(current);
                if (i % 100 == 0) Console.WriteLine(i);
            }
            context.Fact_Matches.AddRange(matchs);
        }

        //auxiliary struct to import facts to fact_table
        private struct Fact
        {
            public DateTime Date { get; set; }
            public int DateID { get; private set; }
            public string HomeTeam { get; set; }
            public int HomeTeamID { get; private set; }
            public string AwayTeam { get; set; }
            public int AwayTeamID { get; private set; }
            public int HomeScore { get; set; }
            public int AwayScore { get; set; }
```

```
            public string Tourn { get; set; }
            public int TournID { get; private set; }
            public string City { get; set; }
            public string Country { get; set; }
            public string Stadium { get; set; }
            public int LocID { get; private set; }
            public string Assistant1 { get; set; }
            public int? Assistant1ID { get; private set; }
            public string Assistant2 { get; set; }
            public int? Assistant2ID { get; private set; }
            public string Referee { get; set; }
            public int? RefereeID { get; private set; }
            public bool Wooms { get; set; }
            public bool neutral { get; set; }

            public void FindIDs(footbal_StarEntities context)
            {
                DateTime temp = Date;
                DateID = context.Date.Where(m => m.Year == temp.Year && m.Month ==
temp.Month && temp.Day == m.Day).Select(m => m.DateID).First();
                string tempStr = HomeTeam;
                HomeTeamID = context.Team.Where(m => m.TeamName == tempStr).Select(m =>
m.TeamID).First();
                tempStr = AwayTeam;
                AwayTeamID = context.Team.Where(m => m.TeamName == tempStr).Select(m =>
m.TeamID).First();
                tempStr = Tourn;
                TournID = context.Tournament.Where(m => m.TournamentName ==
tempStr).Select(m => m.TournamentID).First();
                string tempCountry = Country;
                string tempCity = City;
                string tempStadium = Stadium;
                LocID = context.Location.Where(m => m.CountryName == tempCountry &&
m.City == tempCity && m.Stadium == tempStadium).Select(m => m.LocationID).First();
                if (!string.IsNullOrEmpty(Assistant1))
                {
                    tempStr = Assistant1;
                    Assistant1ID = context.Assistant.Where(m => m.AssitantName ==
tempStr).Select(m => m.AssistantID).First();
                }
                if (!string.IsNullOrEmpty(Assistant2))
                {
                    tempStr = Assistant2;
                    Assistant2ID = context.Assistant.Where(m => m.AssitantName ==
tempStr).Select(m => m.AssistantID).First();
                }
                if (!string.IsNullOrEmpty(Referee))
                {
                    tempStr = Referee;
                    RefereeID = context.Referee.Where(m => m.RefereeName ==
tempStr).Select(m => m.RefereeID).First();
                }
                neutral = !(HomeTeam == Country || AwayTeam == Country);
            }
        }


        //Auxiliary struct to import locations to table
        private struct Loc
        {
            public string Country { get; set; }
            public string City { get; set; }
            public string Stadium { get; set; }
        }
```

```
//Clearing names from trash
private static string ClearName(string team)
{
    team = team.Replace("\"rn\"", "");
    team = team.Replace("\"", "");
    return team.Replace(">", "");
}


//From str monthes to int
private static int MonthToInt(string month)
{
    switch (month)
    {
        case "Jan":
        case "January": return 1;
        case "Feb":
        case "February": return 2;
        case "Mar":
        case "March": return 3;
        case "Apr":
        case "April": return 4;
        case "May": return 5;
        case "Jun":
        case "June": return 6;
        case "Jul":
        case "July": return 7;
        case "Aug":
        case "August": return 8;
        case "Sep":
        case "Sept":
        case "September": return 9;
        case "Oct":
        case "October": return 10;
        case "Nov":
        case "November": return 11;
        case "Dec":
        case "December": return 12;
        default: return 0;
    }
}

//Info about world cup countries
private static string getWorldCupHost(int year, string city)
{
    switch (year)
    {
        case 1930: return "Uruguay";
        case 1934: return "Italy";
        case 1938: return "France";
        case 1950: return "Brazil";
        case 1954: return "Switzerland";
        case 1958: return "Sweden";
        case 1962: return "Chili";
        case 1966: return "England";
        case 1970: return "Mexico";
        case 1974: return "Germany FR";
        case 1978: return "Argentina";
        case 1982: return "Spain";
        case 1986: return "Mexico";
        case 1990: return "Italy";
        case 1994: return "USA";
        case 1998: return "France";
        case 2002: return JapanOrKorea(city);
```

```
                case 2006: return "Germany";
                case 2010: return "South Africa";
                case 2014: return "Brazil";
                default: return "";
            }
        }

        //Info about world cup 2002 country
        private static string JapanOrKorea(string city)
        {
            string[] KoreaCities2002 = { "Seoul", "Ulsan", "Busan", "Gwangju", "Suwon",
"Daegu", "Jeonju", "Jeju", "Incheon", "Daejeon" };
            foreach (string koreaCity in KoreaCities2002)
            {
                if (city.Contains(koreaCity)) return "South Korea";
            }
            return "Japan";
        }

        //Parse str date to three ints
        private static (int, int, int) StrToDateTime(string strDat)
        {
            string day, month, year;
            day = month = year = string.Empty;
            int i = 0;
            do
            {
                day += strDat[i];
                i++;
            } while (Char.IsDigit(strDat[i]) && i < strDat.Length);
            do
            {
                month += strDat[i];
                i++;
            } while (Char.IsLetter(strDat[i]) && i < strDat.Length);
            do
            {
                year += strDat[i];
                i++;
            } while (Char.IsDigit(strDat[i]) && i < strDat.Length);
            return (Convert.ToInt32(year.Trim()), MonthToInt(month.Trim()),
Convert.ToInt32(day.Trim()));
        }
    }
}
```

Контрольні запитання

1. **Дайте визначення ETL.**

ETL – Extract, Transform, Load. Процес який включає в себе вилучення даних із зовнішніх джерел; їх трансформація та очищення, щоб вони відповідали потребам бізнес-моделі; та завантаження їх у сховище даних

2. **На чому базується процес ETL?**

1. Вилучення даних;

2. Перетворення даних;

3. завантаження даних.

Це три процеси, на яких базується ETL.

3. **Переваги та недоліки схем «зірка» та сніжинка.**

Переваги схеми «зірка»:

- ✓ Має просту конструкцію;
- ✓ Запит виконується швидше;
- ✓ Використовує прості запити;
- ✓ Легче зрозуміти.

Недоліки схеми «зірка»:

- Має більше зайвих даних;
- Витрачається більше місця;
- Структура даних денормалізована.

Переваги схеми «сніжинка»:

- ✓ Структура даних нормалізується;
- ✓ Містить менше зайвих даних;
- ✓ Витрачається менше місця.

Недоліки схеми «сніжинка»:

- Має складну конструкцію;
- Запит виконується повільніше;
- Використовує складні запити.

2. **Підходи до створення багатовимірних моделей**

Існують два основні підходи до проектування систем баз даних: низхідний і висхідний. При висхідному підході робота починається з самого нижнього рівня атрибутів (тобто властивостей суті і зв'язків), які на основі аналізу

існуючих між ними зв'язків групуються у відносини, що представляють типи суті і зв'язку між ними. Нормалізація передбачає ідентифікацію необхідних атрибутів з подальшим створенням з них нормалізованих таблиць, заснованих на функціональних залежностях між цими атрибутами. Висхідний підхід найбільшою мірою прийнятний для проектування простих баз даних з відносно невеликою кількістю атрибутів.