



Sri Lanka Institute of Information Technology

Assignment 1

IT3021 - Data Warehousing and Business Intelligence 2022

IT20183554

Chandrasena M.C

Data set selection

I have selected a dataset containing data about Indian Premier League (IPL) matches. Which contains data of each ball played in IPL. The link to the dataset is given below.

<https://www.kaggle.com/datasets/patrickb1912/ipl-complete-dataset-20082020?select=IPL+Matches+2008-2020.csv>

Data from 2013 to 2017 was selected from this data set and have created extra tables considering the relationships. The dataset is transformed in order to analyze playe wise performance.

Preparation of data sources

Initially all the data were in 'csv' format. They were converted into different types of data sources as, one database backup file, two 'csv' files and a text file. Following are the tables which were separated into different sources;

1. Database(.bak)
 - 1.1. Player
 - 1.2. Venue
 - 1.3. OutType
 - 1.4. ExtraType
 - 1.5. BallingStyle
 - 1.6. BattingStyle
 - 1.7. BallByBall
 - 1.8. BallData
2. Comma Seperated Values(.csv)
 - 2.1. Country
 - 2.2. Team
3. Text(.txt)
 - 3.1. VenueAddress

Data Source Type	Source Name	Column Name	Data Type	Description
Database File (.bak)	dbo. BallByBall	BallDataID	int	Includes facts of the IPL matches ball by ball.
		BallDataID	int	
		TeamBattingID	int	
		TeamBowlingID	int	
		StrikerID	int	
		NonStrikerID	int	
		RunsScored	int	
		ExtraTypeID	int	
		ExtraRuns	int	
		OutTypeID	int	
		OutPlayerID	int	
		IsBowlerWicket	int	
		BowlerID	int	
		FielderID	int	
		MatchDate	datetme	
		VenueID	int	
	dbo.BallData	BallDataID	int	

	MatchNo	int	Contains data about the balls in every match as a hierarchy. Ex: Third ball of second over of the first innings, of the fifth match.
	InningsNo	int	
	OverNo	int	
	BallNo	int	
dbo.Venue	VenueID	int	Contains details of grounds where matches are played.
	VenueName	nvarchar(255)	
dbo.ExtraType	ExtraTypeID	int	Contains data types extras.
	ExtraType	nvarchar(255)	
dbo.OutType	OutTypeID	int	Contains data about types of wickets.
	OutType	nvarchar(255)	
dbo.Player	PlayerID	int	Contains details of players.
	PlayeName	nvarchar(255)	
	PlayerNameInitials	nvarchar(255)	
	CountryID	int	
	BattingStyle	int	
	BowlingStyleID	int	
dbo.BattingStyle	BattingStyleID	int	Contains details of batting styles.
	BattingStyle	nvarchar(255)	
dbo.BowlingStyle	BowlingStyleID	int	

		BowlingStyle	nvarchar(255)	Contains details of batting styles.
CSV File	Country.csv	CountryID	int	Contains details of countries of players.
		CountryName	nvarchar(255)	
	Team.csv	TeamID	int	Contains details of teams of the tournament.
		TeamName	nvarchar(255)	
Text file	VenueAddress.txt	VenueAddressID	int	Contains details addresses of the venues (grounds)
		CityName	nvarchar(255)	
		CountryName	nvarchar(255)	

Solution architecture

Data Sources

Data Sources represent the sources which were used to get data. There are three types of sources as csv, text and .bak which represents comma separated files, text files and database files respectively

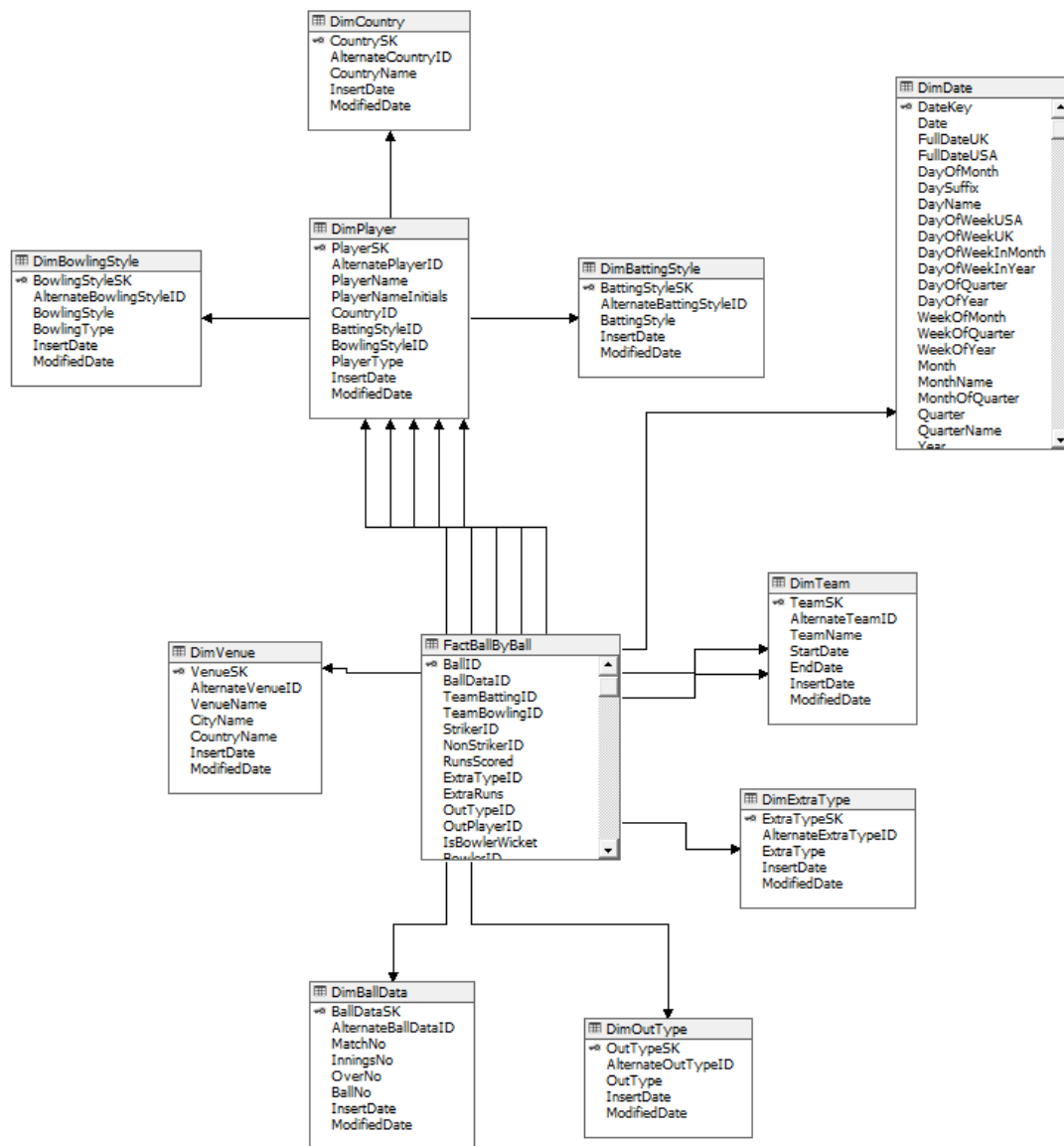
Staging Area

This level represent creating staging level tables using the data which were obtained by different data sources.

Data Warehouse

Here, data in the staging area are transformed and loaded into the data warehouse as facts and dimensions which is then used for Business Intelligence purposes.

Data warehouse design & development



Above diagram shows how the dimension tables and fact table was combined.

Following were considered when developing the data warehouse dimensional model;

- Snowflake schema type was used.
- **Dimensions**
 - Hierarchical dimensions

1. Venue – Country name – City name – Venue name
2. BowlingStyle – Bowling Type – Bowling Style
3. BallData – Match number – Innings number – Over number – Ball number
4. Date

➤ **Slowly changing dimensions**

1. Team – Team name

- **Fact Table**

➤ **BallByBall**

This table consists of 12 foreign key columns which are connected to the dimensions of the model.

- **Assumptions**

- Since the names of the teams are changed by owners when needed, Team was considered as a slowly changing dimension to track the details of the team names.

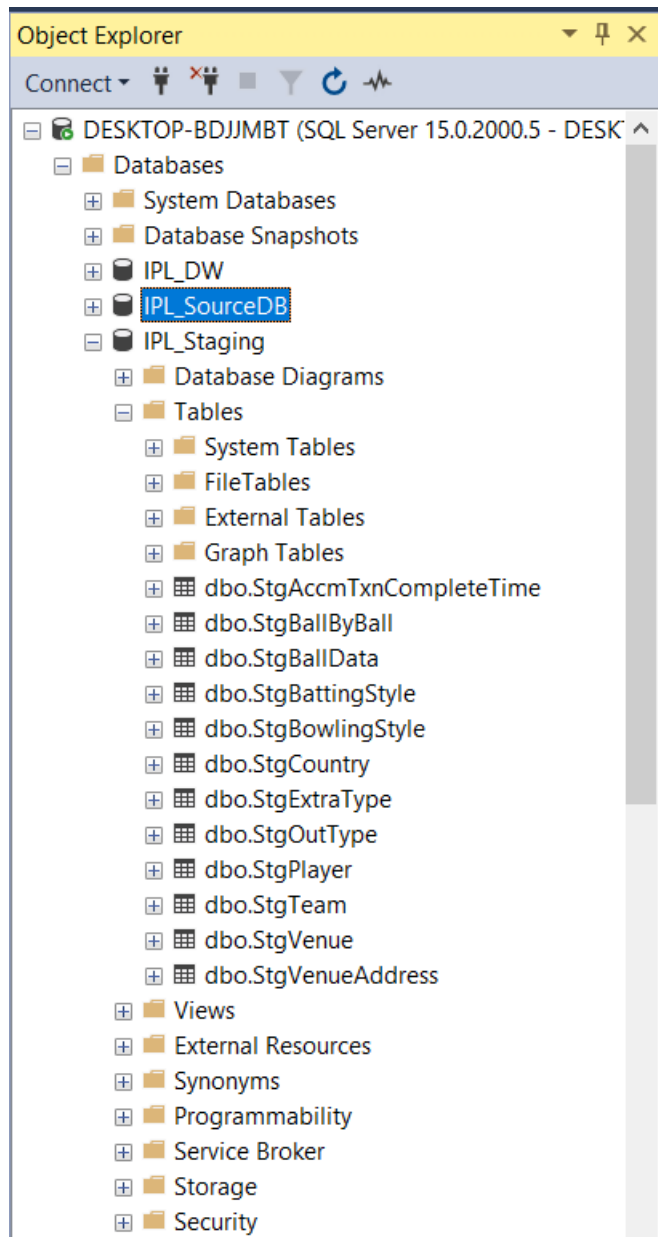
ETL development

Extract

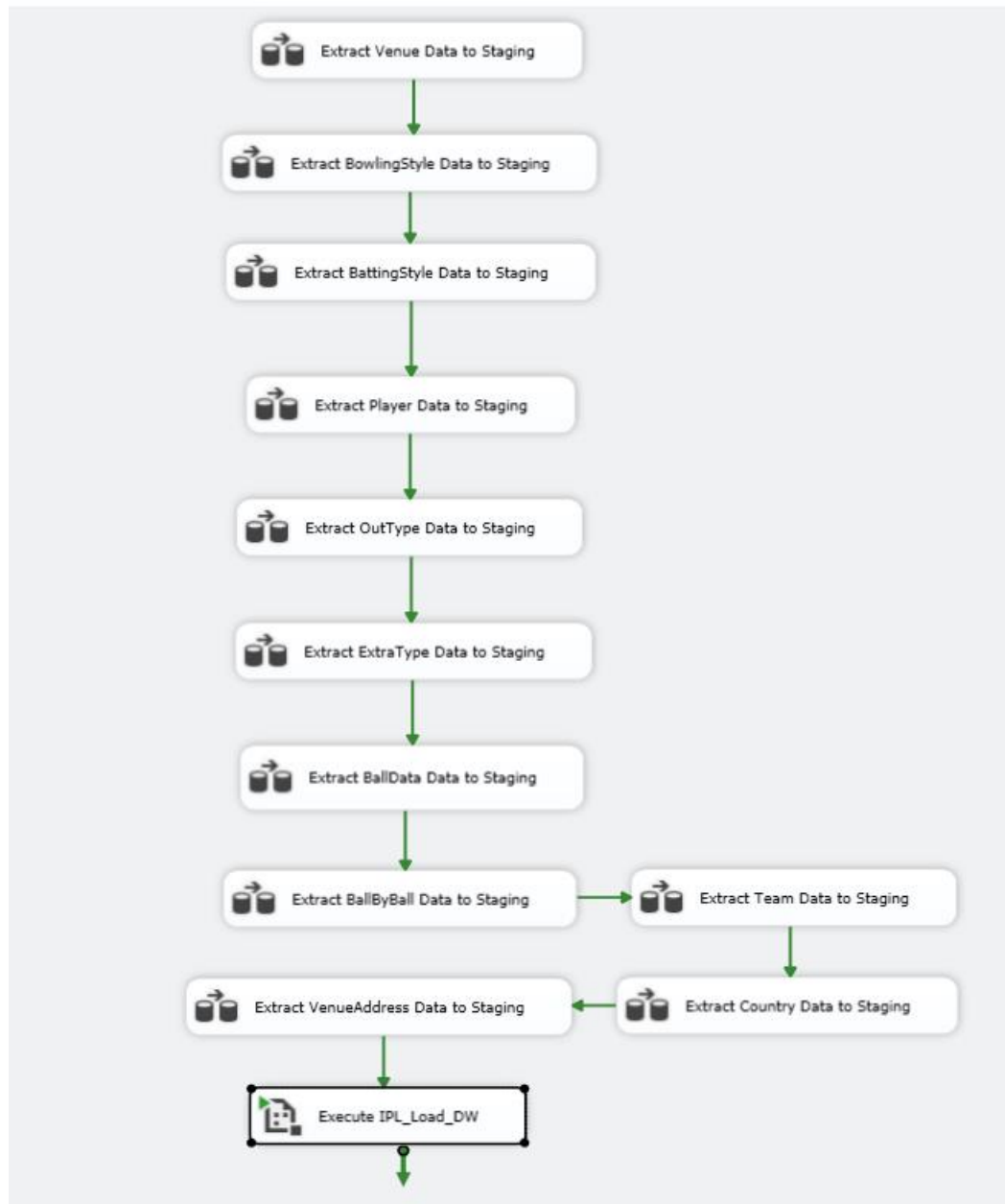
First, all the data which mentioned in the Preparation of Data Sources step were imported to the staging database (IPL_Staging) by using relevant connections and the sources.

Below image shows the tables of the staging database;

SSMS staging database (IPL_Staging)

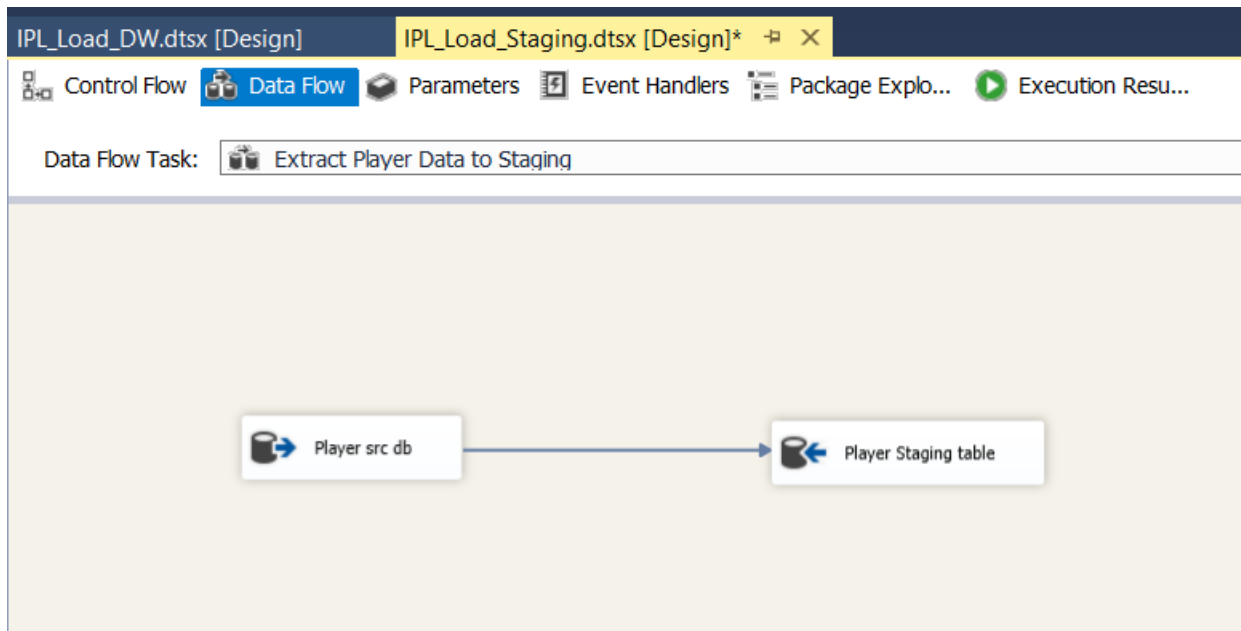


Control flow of extraction

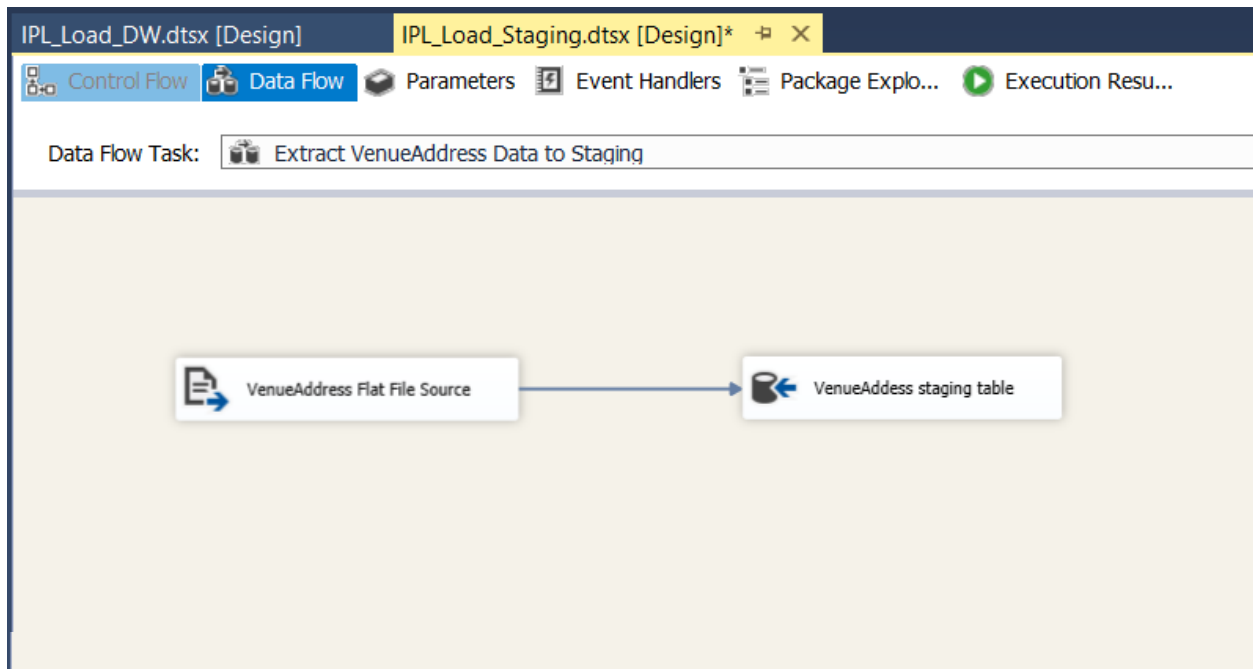


Screenshots of some data flows are given below;

Player data flow

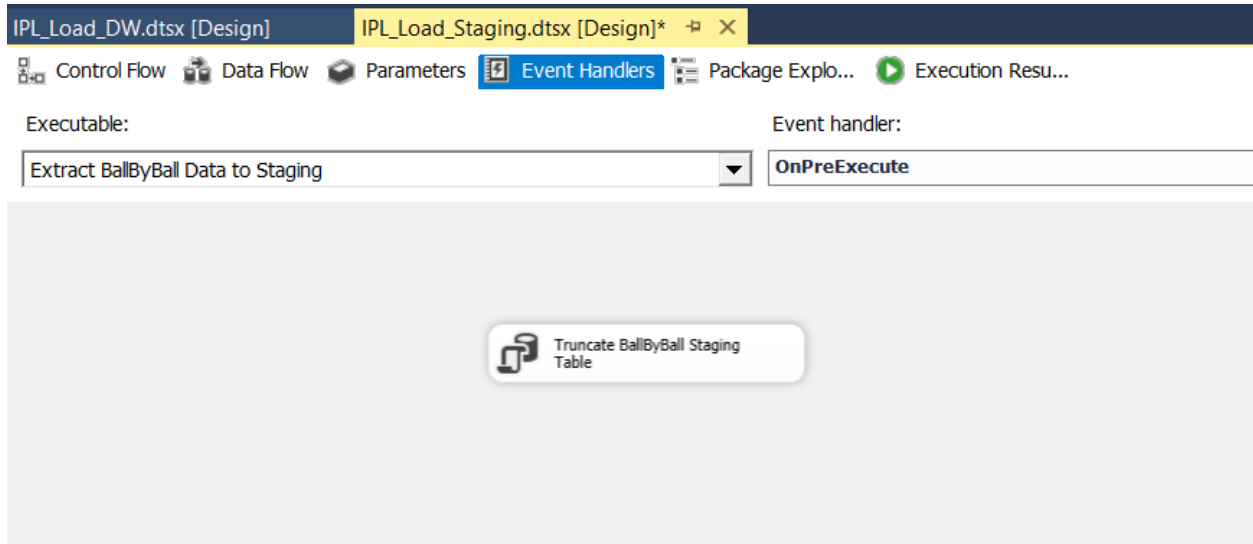


Country data flow

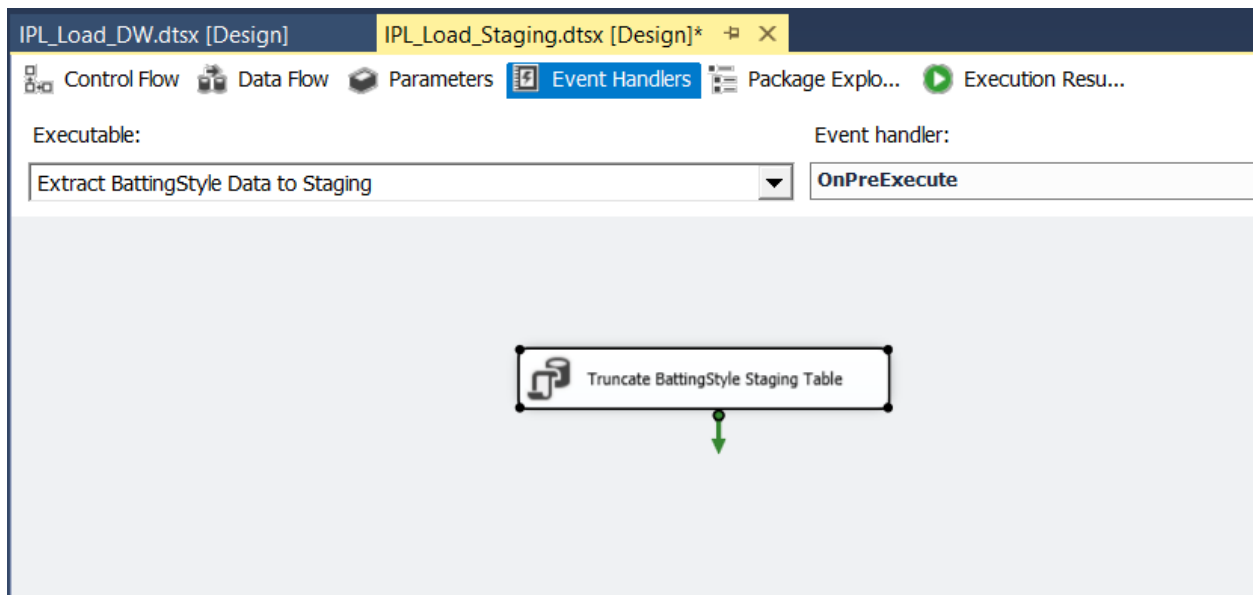


Screenshots of some event handlers are given below;

Truncate BallByBall Staging



Truncate BattingStyle Staging



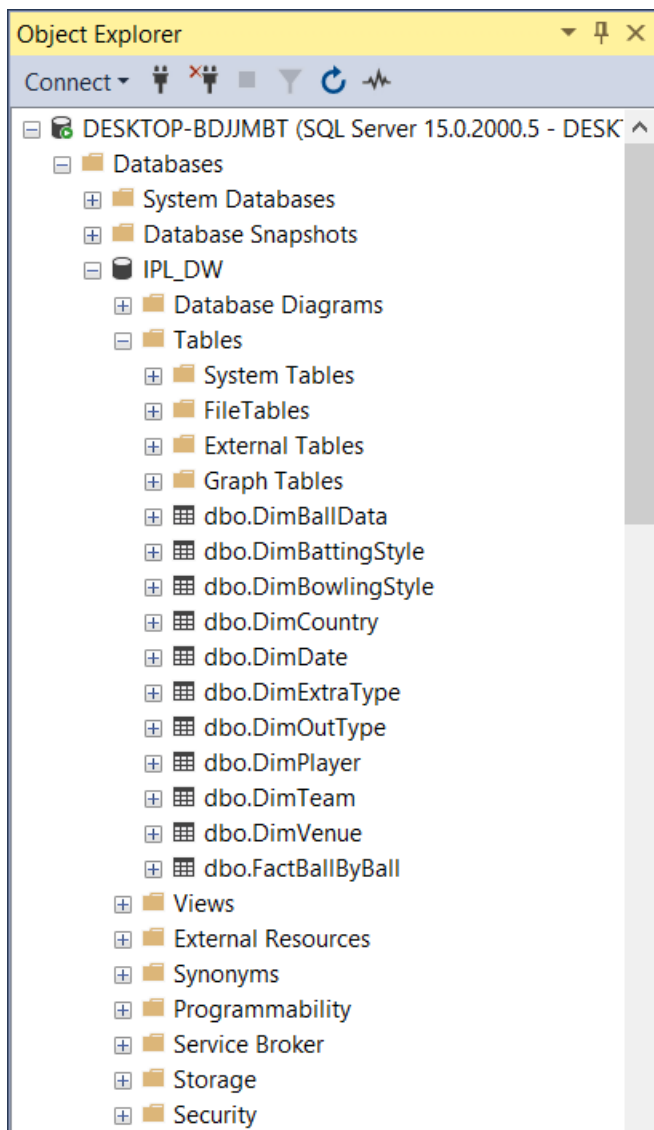
Transform and load

Next the data in the staging area were transformed and loaded in to the data warehouse (IPL_DW). First the dimension tables and the fact table was created and the data were loaded to the in the relevant order.

Tasks such as merge join, lookup, derived columns, and sort were used in transforming and loading data to data warehouse.

Below image shows the tables of the staging database;

SSMS data warehouse (IPL_DW)



Sql Query to create fact table;

```
BallByBallDW.sql - not connected X SQLQuery11.sql - D...BDJMBT\deli (66))* SQLQuery7.sq
drop table if exists FactBallByBall;
create table FactBallByBall
(
    BallID int primary key,
    BallDataID int foreign key references DimBallData(BallDataSK),
    TeamBattingID int foreign key references DimTeam(TeamSK),
    TeamBowlingID int foreign key references DimTeam(TeamSK),
    StrikerID int foreign key references DimPlayer(PlayerSK),
    NonStrikerID int foreign key references DimPlayer(PlayerSK),
    RunsScored int,
    ExtraTypeID int foreign key references DimExtraType(ExtraTypeSK),
    ExtraRuns int,
    OutTypeID int foreign key references DimOutType(OutTypeSK),
    OutPlayerID int foreign key references DimPlayer(PlayerSK),
    IsBowlerWicket int,
    BowlerID int foreign key references DimPlayer(PlayerSK),
    FielderID int foreign key references DimPlayer(PlayerSK),
    MatchDate int foreign key references DimDate(DateKey),
    VenueID int foreign key references DimVenue(VenueSK),
    InsertDate DateTime,
    ModifiedDate DateTime,
    accm_txn_create_time DateTime,
    accm_txn_complete_time DateTime,
    txn_process_time_hours int
)
```

Screenshots of some sql procedures are given below;

Procedure for dimPlayer

```
PlayerProcedure.sql...BDJIMBT\deli (64)  X  BallByBallDW.sql - not connected  SQLQuery11.sql - D...BDJIMBT\deli (66)*  SQLQuery7.sql - DE...BDJIMBT\deli (63))*  TeamProcedure.sql...BDJIMBT\deli (52))

CREATE PROCEDURE dbo.UpdatePlayer
    @PlayerID int,
    @PlayerName nvarchar(50),
    @PlayerNameInitials nvarchar(50),
    @CountryID int,
    @BattingStyleID int,
    @BowlingStyleID int,
    @PlayerType nvarchar(25)
AS
BEGIN
    if not exists (select PlayerSK
    from dbo.DimPlayer
    where AlternatePlayerID = @PlayerID)
    BEGIN
        insert into dbo.DimPlayer
        (AlternatePlayerID, PlayerName, PlayerNameInitials, CountryID, BattingStyleID, BowlingStyleID, PlayerType, InsertDate, ModifiedDate)
        values
        (@PlayerID, @PlayerName, @PlayerNameInitials, @CountryID, @BattingStyleID, @BowlingStyleID, @PlayerType, GETDATE(), GETDATE())
    END;
    if exists (select PlayerSK
    from dbo.DimPlayer
    where AlternatePlayerID = @PlayerID)
    BEGIN
        update dbo.DimPlayer
        set PlayerName = @PlayerName,
        PlayerNameInitials = @PlayerNameInitials,
        CountryID = @CountryID,
        BattingStyleID = @BattingStyleID,
        BowlingStyleID = @BowlingStyleID,
        PlayerType = @PlayerType,
        ModifiedDate = GETDATE()
        where AlternatePlayerID = @PlayerID and (PlayerName != @PlayerName or PlayerNameInitials != @PlayerNameInitials or CountryID != @CountryID or BattingStyleID != @BattingStyleID or BowlingStyleID != @BowlingStyleID)
    END;
END;
```

Procedure for dimVenue

```
VenueProcedure.sql...BDJIMBT\deli (65)  X  PlayerProcedure.sql...BDJIMBT\deli (64))  BallByBallDW.sql - not connected  SQLQuery11.sql - D...BDJIMBT\deli (66)*


CREATE PROCEDURE dbo.UpdateVenue
    @VenueID int,
    @VenueName nvarchar(100),
    @CityName nvarchar(50),
    @CountryName nvarchar(50)
AS
BEGIN
    if not exists (select VenueSK
    from dbo.DimVenue
    where AlternateVenueID = @VenueID)
    BEGIN
        insert into dbo.DimVenue
        (AlternateVenueID, VenueName, CityName, CountryName, InsertDate, ModifiedDate)
        values
        (@VenueID, @VenueName, @CityName, @CountryName, GETDATE(), GETDATE())
    END;
    if exists (select VenueSK
    from dbo.DimVenue
    where AlternateVenueID = @VenueID)
    BEGIN
        update dbo.DimVenue
        set VenueName = @VenueName,
        CityName = @CityName,
        CountryName = @CountryName,
        ModifiedDate = GETDATE()
        where AlternateVenueID = @VenueID and (VenueName != @VenueName or CityName != @CityName or CountryName != @CountryName)
    END;
END;
```

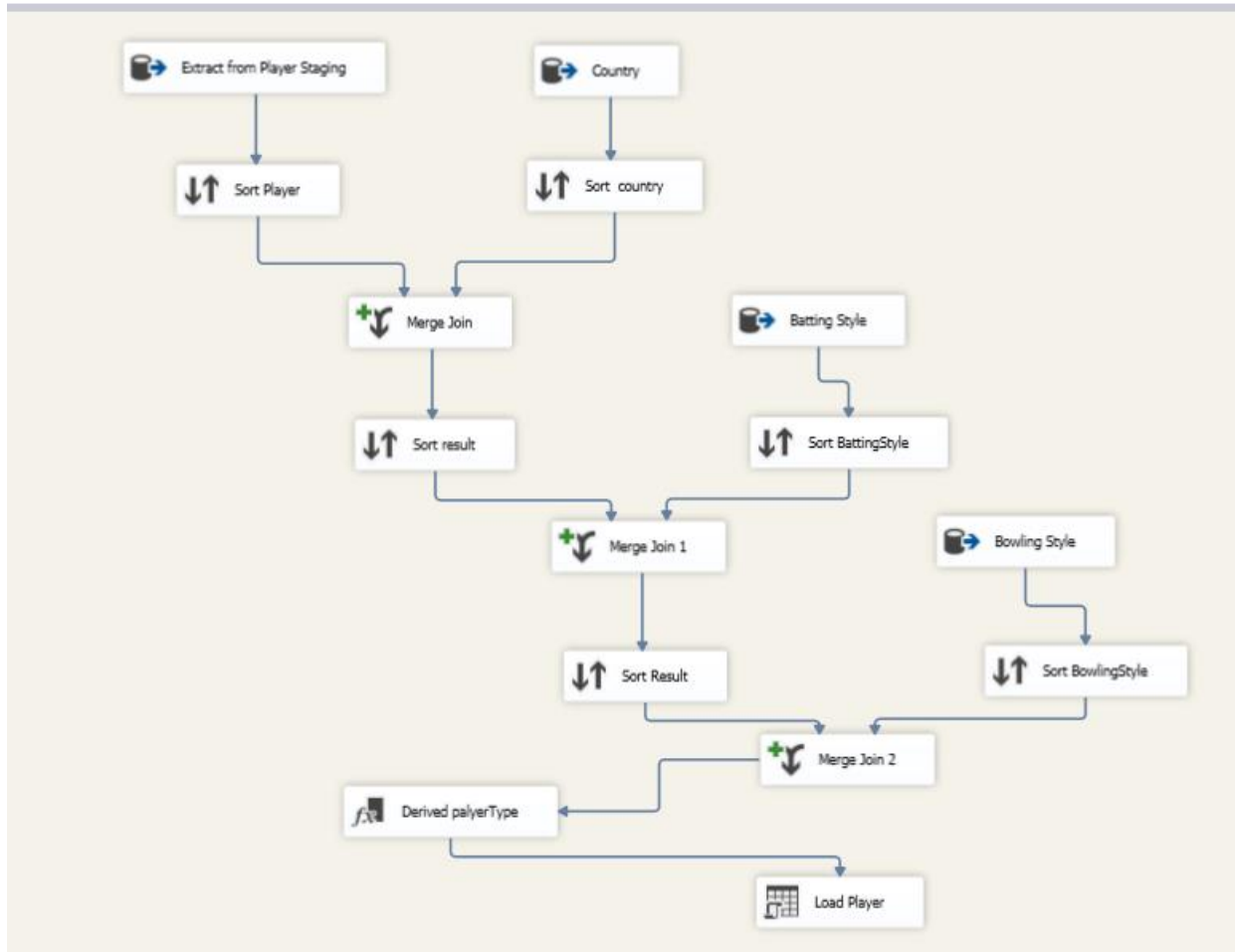
Control flow extraction



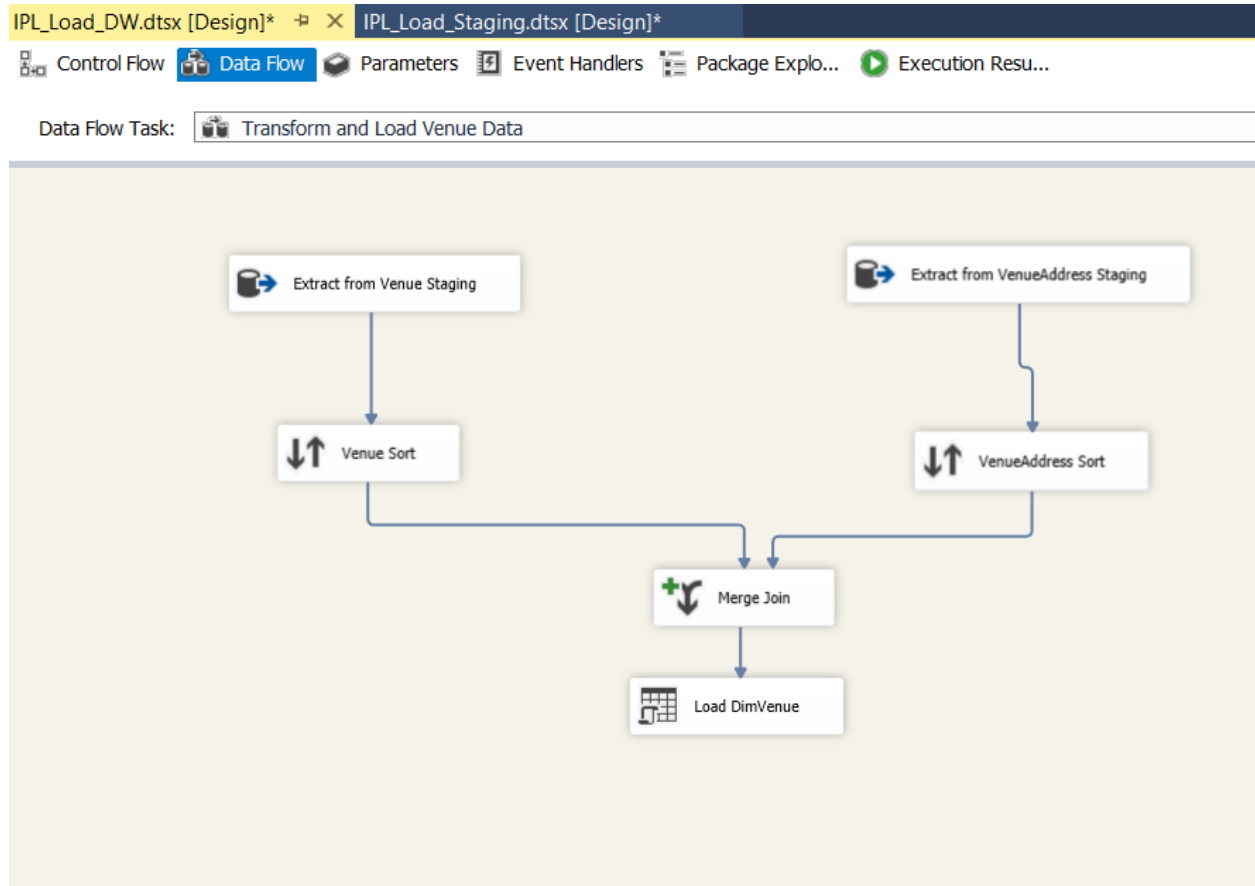
Screenshots of some data flows are given below;

DimPlayer transform and load

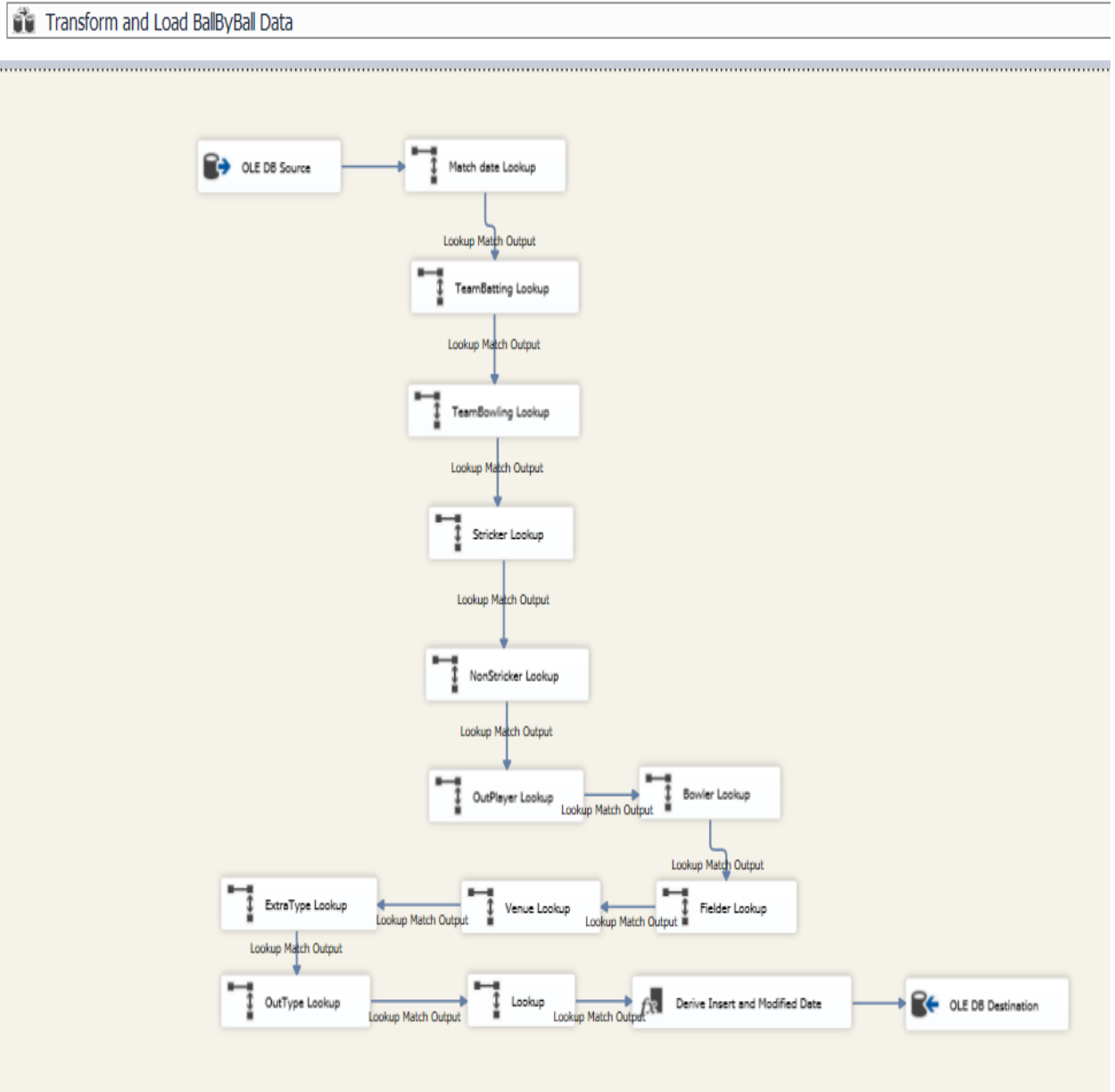
Data Flow Task:  Transform and Load Player Data



Dim Venue transform and load



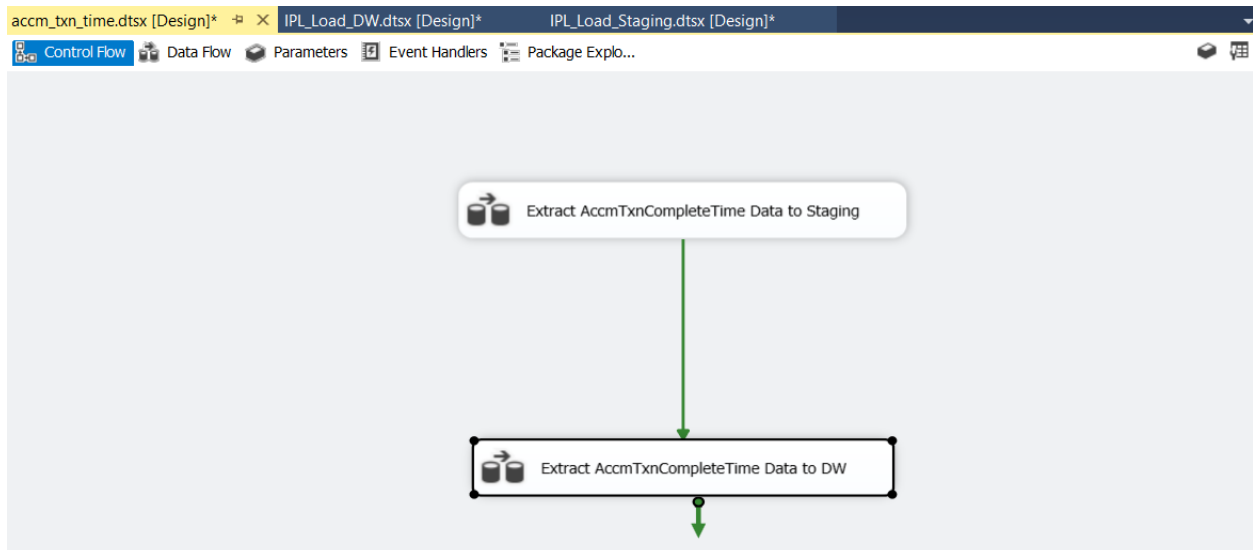
Fact BallByBall transform and load



ETL development – Accumulating fact tables

An external csv data source was used for this step and relevant coulombs were created in fact table. Data was transformed and loaded to these fields using separate ETL process.

Control flow extraction



Transform and load to Fact BallByBall

