All T20 Internationals Dataset (2005 - 2023)

Strategic Upgrade to Non-Relational Database: A Comprehensive Assessment for T20 Cricket Data Management | Unit 8

Avinash Bunga

Information Systems and Business Analytics

Park University CIS622DLAF2P2023

Data Architecture for Business Analytics

Professor: Gulnoza Khakimova

Dec 10, 2023

Introduction: Strategic Upgrade to Non-Relational Database: A Comprehensive Assessment for T20 Cricket Data Management

In this assessment, I embarked on a comprehensive journey to evaluate and prepare for transitioning our T20 cricket database to a non-relational system. This project represents a significant endeavor on my part to enhance the way we manage and analyze cricket data. It involved the meticulous creation of a data dictionary and an ERD, the development of robust SQL code, and a thorough evaluation of the potential benefits and challenges associated with moving to a non-relational database. Each step of this process, driven by my effort and analysis, aimed to ensure a seamless and effective transition, aiming to elevate our cricket data analytics to new heights of efficiency and insight.

Data Dictionary

The Data Dictionary for the "All T20 Internationals Dataset (2005 - 2023)" is a comprehensive guide for understanding the multifaceted data related to T20 cricket. It includes a detailed description of attributes for three critical data files: 't20i_Matches_Data', 't20i_Batting_Card', and 't20i_Bowling_Card', each contributing to a rich analytical framework for match and player performance analysis. A unique 'Match ID' attribute across these files ensures seamless integration and comparative analysis. This dictionary is fundamental for stakeholders like analysts, teams, and enthusiasts to navigate through complex data, enabling informed decisions and insights into the evolving trends of international T20 cricket.

Below are the three key tables that encapsulate the diverse data points of our T20 cricket database:

Data Dictionary 1: t20i_Matches_Data

Field Name in Original Source	Field Name in Database	Data Type	Description	Required ?	Accept NULL?
T20I Match No	MatchNo	Integer	Unique number representing each T20I match	Yes	No
Match ID	MatchID	Integer	Unique identifier for each match	Yes	No
Match Name	MatchName	Text	Name of the match	Yes	No
Series ID	SeriesID	Integer	Unique identifier for each series	Yes	No
Series Name	SeriesName	Text	Name of the cricket series	Yes	No
Match Date	MatchDate	Date	Date of the match	Yes	No
Match Format	MatchFormat	Text	Format of the match (e.g., T20)	Yes	No
Team1 ID	Team1ID	Integer	Unique identifier for the first team	Yes	No
Team1 Name	Team1Name	Text	Name of the first team	Yes	No
Team1 Captain	Team1Captain	Text	Name of the captain of the first team	Yes	No
Team1 Runs Scored	Team1Runs	Integer	Total runs scored by the first team	Yes	Yes
Team1 Wickets Fell	Team1Wickets	Integer	Total wickets fallen for the first team	Yes	Yes
Team1 Extras Rec	Team1Extras	Integer	Extra runs received by the first team	Yes	Yes
Team2 ID	Team2ID	Integer	Unique identifier for the second team	Yes	No
Team2 Name	Team2Name	Text	Name of the second team	Yes	No
Team2 Captain	Team2Captain	Text	Name of the captain of the second team	Yes	Yes
Team2 Runs Scored	Team2Runs	Integer	Total runs scored by the second team	Yes	Yes
Team2 Wickets Fell	Team2Wickets	Integer	Total wickets fallen for the second team	Yes	Yes

Team2 Extras Rec	Team2Extras	Integer	Extra runs received by the second team	Yes	Yes
Match Venue (Stadium)	VenueStadium	Text	Name of the stadium where the match took place	Yes	Yes
Match Venue (City)	VenueCity	Text	City where the match took place	Yes	No
Match Venue (Country)	VenueCountry	Text	Country where the match took place	Yes	No
Umpire 1	Umpire1	Text	Name of the first umpire	Yes	Yes
Umpire 2	Umpire2	Text	Name of the second umpire	Yes	Yes
Match Referee	Referee	Text	Name of the match referee	Yes	Yes
Toss Winner	TossWinner	Text	Team that won the toss	Yes	No
Toss Winner Choice	TossChoice	Text	Decision taken by the toss-winning team (e.g., bat or bowl)	Yes	Yes
Match Winner	MatchWinner	Text	Team that won the match	Yes	Yes
Match Result Text	MatchResult	Text	Textual representation of the match result	Yes	Yes
MOM Player	ManOfTheMatch	Text	Player awarded "Man of the Match"	Yes	Yes
Team1 Playing 11	Team1Playing11	Text Array	List of players in the playing 11 of the first team	Yes	Yes
Team2 Playing 11	Team2Playing11	Text Array	List of players in the playing 11 of the second team	Yes	Yes
Debut Players	DebutPlayers	Text Array	List of players making their debut in the match	Yes	Yes

Data Dictionary 2: t20i_Batting_Card

Field Name in Original Source	Field Name in Database	Data Type	Description	Required?	Accept NULL?
Match ID	MatchID	Integer	Unique identifier for each match	Yes	No
innings	Innings	Integer	The innings number (1 or 2)	Yes	No
team	Team	Text	Name of the team the batsman belongs to	Yes	No
batsman	BatsmanName	Text	Name of the batsman	Yes	No
battingStyle	BattingStyle	Text	Style of batting (e.g., right-hand bat, left-hand bat)	Yes	No
runs	RunsScored	Integer	Number of runs scored by the batsman	Yes	No
balls	BallsFaced	Integer	Number of balls faced by the batsman	Yes	No
fours	Fours	Integer	Number of fours hit by the batsman	Yes	No
sixes	Sixes	Integer	Number of sixes hit by the batsman	Yes	No
strikeRate	StrikeRate	Float	Strike rate of the batsman	Yes	No
isOut	IsOut	Boolean	Whether the batsman got out or not	Yes	No
wicketType	WicketType	Text	Type of wicket (e.g., bowled, caught)	Yes	Yes
fielders	Fielders	Text Array	List of fielders involved in the wicket	Yes	Yes
bowler	Bowler	Text	Name of the bowler who took the wicket	Yes	Yes

Data Dictionary 3: t20i_Bowling_Card

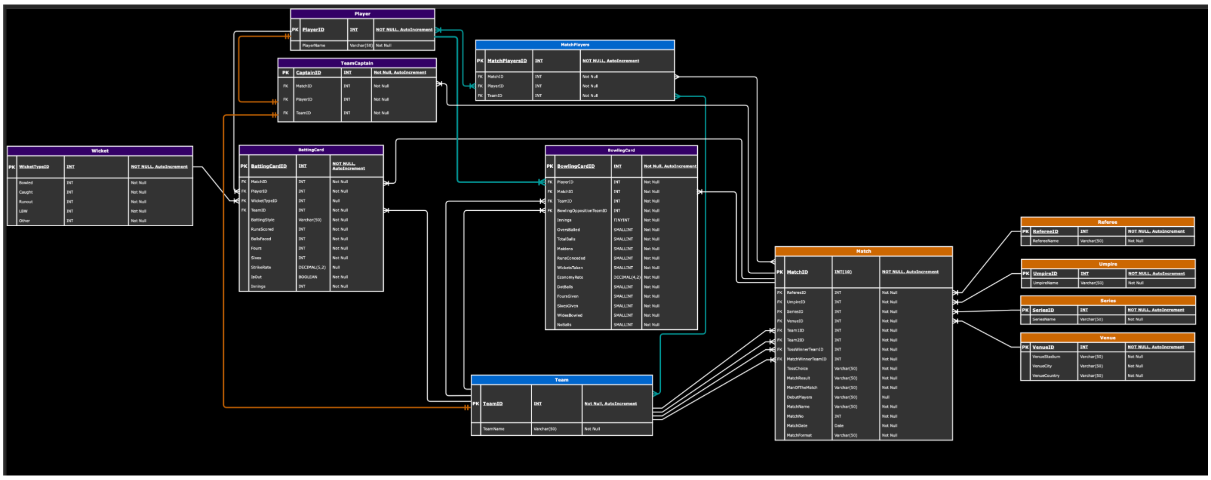
Field Name in Original Source	Field Name in Database	Data Type	Description	Required?	Accept NULL?
Match ID	MatchID	Integer	Unique identifier for each match	Yes	No
innings	Innings	Integer	The innings number (1 or 2)	Yes	No
team	Team	Text	Bowling team's name	Yes	No
opposition	Opposition	Text	Name of the team they are bowling against	Yes	No
name	BowlerName	Text	Name of the bowler	Yes	No
overs	OversBalled	Float	Total overs bowled by the bowler	Yes	No
balls	TotalBalls	Integer	Total balls bowled by the bowler in number format	Yes	No
maidens	Maidens	Integer	Total maiden overs bowled by the bowler	Yes	No
conceded	RunsConceded	Integer	Runs conceded by the bowler	Yes	No
wickets	WicketsTaken	Integer	Total wickets taken by the bowler	Yes	No
economy	EconomyRate	Float	Economy rate of the bowler	Yes	No
dots	DotBalls	Integer	Number of dot balls bowled	Yes	No
fours	FoursGiven	Integer	Number of boundary fours given by the bowler	Yes	No
sixes	SixesGiven	Integer	Number of sixes given by the bowler	Yes	No
wides	WidesBalled	Integer	Number of wide balls bowled	Yes	No
noballs	NoBalls	Integer	Number of no balls bowled	Yes	No

GitHub Link

Entity Relationship Diagram (ERD)

In Unit 3, we embarked on the detailed construction of an Entity-Relationship Diagram (ERD) for the "All T20 Internationals Dataset (2005 - 2023)," ensuring adherence to the 3rd Normal Form for database normalization. This rigorous process involved delineating 12 distinct tables tailored to reflect the complexities of T20 cricket data while optimizing for data integrity and query efficiency. The ERD, a cornerstone of our database architecture, was meticulously structured to facilitate future updates, eliminate data redundancies, and uphold a robust framework for data analysis. For an in-depth look at the ERD and the normalization process. GitHub Link

Below is the **Physical ERD** that encapsulates our database's structure, showcasing the interconnections and relationships critical for our comprehensive T20 cricket data analysis:



Player

PK	PlayerID	INT	NOT NULL, AutoIncrement
	PlayerName	Varchar(50)	Not Null

TeamCaptain

PK	CaptainID	INT	Not Null, AutoIncrement
FK	MatchID	INT	Not Null
FK	PlayerID	INT	Not Null
FK	TeamID	INT	Not Null

MatchPlayers

PK	MatchPlayersID	INT	NOT NULL, AutoIncrement
FK	MatchID	INT	Not Null
FK	PlayerID	INT	Not Null
FK	TeamID	INT	Not Null

Wicket

PK	WicketTypeID	INT	NOT NULL, AutoIncrement
	Bowled	INT	Not Null
	Caught	INT	Not Null
	Runout	INT	Not Null
	LBW	INT	Not Null
	Other	INT	Not Null

BattingCard

PK	BattingCardID	INT	NOT NULL, AutoIncrement
FK	MatchID	INT	Not Null
FK	PlayerID	INT	Not Null
FK	WicketTypeID	INT	Null
FK	TeamID	INT	Not Null
	BattingStyle	Varchar(50)	Not Null
	RunsScored	INT	Not Null
	BallsFaced	INT	Not Null
	Fours	INT	Not Null
	Sixes	INT	Not Null
	StrikeRate	DECIMAL(5,2)	Null
	IsOut	BOOLEAN	Not Null
	Innings	INT	Not Null

BowlingCard

PK	BowlingCardID	INT	Not Null, AutoIncrement
FK	PlayerID	INT	Not Null
FK	MatchID	INT	Not Null
FK	TeamID	INT	Not Null
FK	BowlingOppositionTeamID	INT	Not Null
	Innings	TINYINT	Not Null
	OversBalled	SMALLINT	Not Null
	TotalBalls	SMALLINT	Not Null
	Maidens	SMALLINT	Not Null
	RunsConceded	SMALLINT	Not Null
	WicketsTaken	SMALLINT	Not Null
	EconomyRate	DECIMAL(4,2)	Not Null
	DotBalls	SMALLINT	Not Null
	FoursGiven	SMALLINT	Not Null
	SixesGiven	SMALLINT	Not Null
	WidesBowled	SMALLINT	Not Null
	NoBalls	SMALLINT	Not Null

Team

PK	TeamID	INT	Not Null, AutoIncrement
	TeamName	Varchar(50)	Not Null

Match

PK	MatchID	INT(10)	NOT NULL, AutoIncrement
FK	RefereeID	INT	Not Null
FK	UmpireID	INT	Not Null
FK	SeriesID	INT	Not Null
FK	VenueID	INT	Not Null
FK	Team1ID	INT	Not Null
FK	Team2ID	INT	Not Null
FK	TossWinnerTeamID	INT	Not Null
FK	MatchWinnerTeamID	INT	Not Null
	TossChoice	Varchar(50)	Not Null
	MatchResult	Varchar(50)	Not Null
	ManOfTheMatch	Varchar(50)	Not Null
	DebutPlayers	Varchar(50)	Null
	MatchName	Varchar(50)	Not Null
	MatchNo	INT	Not Null
	MatchDate	Date	Not Null
	MatchFormat	Varchar(50)	Not Null

Referee

PK	RefereeID	INT	NOT NULL, AutoIncrement
	RefereeName	Varchar(50)	Not Null

Umpire

PK	UmpireID	INT	NOT NULL, AutoIncrement
	UmpireName	Varchar(50)	Not Null

Series

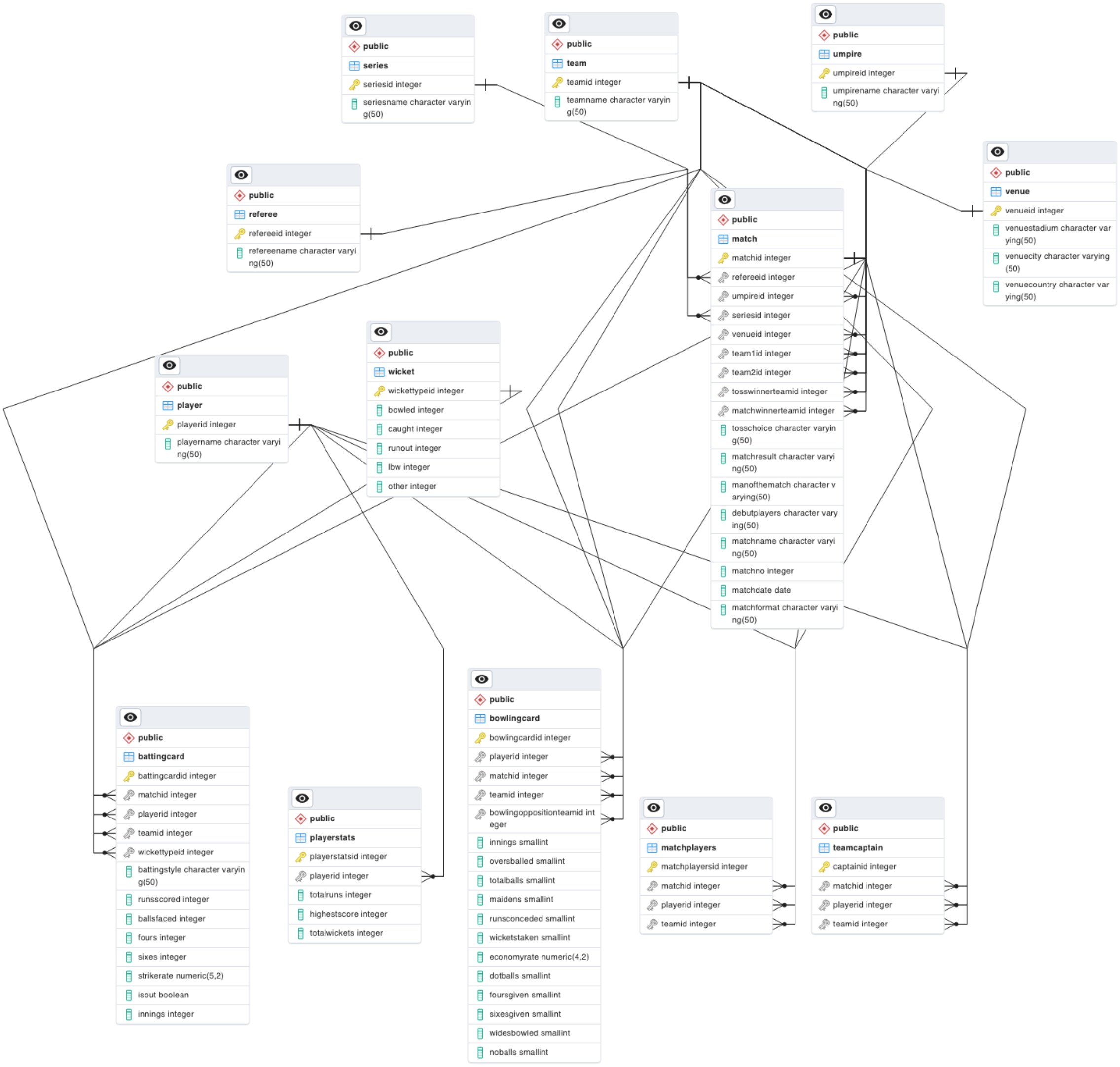
PK	SeriesID	INT	NOT NULL, AutoIncrement
	SeriesName	Varchar(50)	Not Null

Venue

PK	VenueID	INT	NOT NULL, AutoIncrement
	VenueStadium	Varchar(50)	Not Null
	VenueCity	Varchar(50)	Not Null
	VenueCountry	Varchar(50)	Not Null

GitHub Link

In the progression of this project, I meticulously translated our database's conceptual design into a tangible structure using pgAdmin. This step culminated in the creation of an Entity-Relationship Diagram (ERD), which visually encapsulates the intricate relationships and constraints of our tables, ensuring a robust and coherent schema:



This ERD is a testament to the precision with which the database was crafted, offering a snapshot of the relational framework that underpins our comprehensive T20 cricket data management system. GitHub Link

SQL code for database

Building upon the structured design from our ERD, I developed the SQL code necessary to bring our T20 cricket database to life. The SQL script I wrote is the backbone that constructs the database tables, enforces data integrity, and establishes the relationships crucial for our complex queries. Here is a glimpse of the initial SQL code I crafted, which laid the groundwork for our data storage and retrieval system: GitHub Link

```sql
CREATE DATABASE All_T20_Internationals_Dataset_2005_2023;
USE All_T20_Internationals_Dataset_2005_2023;

CREATE TABLE Player (
 PlayerID INT NOT NULL AUTO_INCREMENT,
 PlayerName VARCHAR(50) NOT NULL,
 PRIMARY KEY (PlayerID)
);

CREATE TABLE Team (
 TeamID INT NOT NULL AUTO_INCREMENT,
 TeamName VARCHAR(50) NOT NULL,
 PRIMARY KEY (TeamID)
);

CREATE TABLE Wicket (
 WicketTypeID INT NOT NULL AUTO_INCREMENT,
 Bowled INT NOT NULL,
 Caught INT NOT NULL,
 Runout INT NOT NULL,
 LBW INT NOT NULL,
 Other INT NOT NULL,
 PRIMARY KEY (WicketTypeID)
);

CREATE TABLE MatchPlayers (
 MatchPlayersID INT NOT NULL AUTO_INCREMENT,
 MatchID INT NOT NULL,
 PlayerID INT NOT NULL,
 TeamID INT NOT NULL,
 PRIMARY KEY (MatchPlayersID),
 FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
 FOREIGN KEY (TeamID) REFERENCES Team(TeamID)
);

CREATE TABLE BattingCard (
 BattingCardID INT NOT NULL AUTO_INCREMENT,
 MatchID INT NOT NULL,
 PlayerID INT NOT NULL,
 TeamID INT NOT NULL,
 WicketTypeID INT,
 BattingStyle VARCHAR(50) NOT NULL,
 RunsScored INT NOT NULL,
 BallsFaced INT NOT NULL,
 Fours INT NOT NULL,
 Sixes INT NOT NULL,
 StrikeRate DECIMAL(5,2),
 IsOut BOOLEAN NOT NULL,
 Innings INT NOT NULL,
 PRIMARY KEY (BattingCardID),
 FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
 FOREIGN KEY (TeamID) REFERENCES Team(TeamID),
 FOREIGN KEY (WicketTypeID) REFERENCES Wicket(WicketTypeID)
);

CREATE TABLE BowlingCard (
 BowlingCardID INT NOT NULL AUTO_INCREMENT,
 PlayerID INT NOT NULL,
 MatchID INT NOT NULL,
 TeamID INT NOT NULL,
 BowlingOppositionTeamID INT NOT NULL,
 Innings TINYINT NOT NULL,
 OversBalled SMALLINT NOT NULL,
 TotalBalls SMALLINT NOT NULL,
 Maidens SMALLINT NOT NULL,
 RunsConceded SMALLINT NOT NULL,
 WicketsTaken SMALLINT NOT NULL,
 EconomyRate DECIMAL(4,2) NOT NULL,
 DotBalls SMALLINT NOT NULL,
 FoursGiven SMALLINT NOT NULL,
 SixesGiven SMALLINT NOT NULL,
 WidesBowled SMALLINT NOT NULL,
 NoBalls SMALLINT NOT NULL,
 PRIMARY KEY (BowlingCardID),
 FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
 FOREIGN KEY (TeamID) REFERENCES Team(TeamID),
 FOREIGN KEY (BowlingOppositionTeamID) REFERENCES Team(TeamID)
);
```

```sql
CREATE TABLE Referee (
 RefereeID INT NOT NULL AUTO_INCREMENT,
 RefereeName VARCHAR(50) NOT NULL,
 PRIMARY KEY (RefereeID)
);

CREATE TABLE Umpire (
 UmpireID INT NOT NULL AUTO_INCREMENT,
 UmpireName VARCHAR(50) NOT NULL,
 PRIMARY KEY (UmpireID)
);

CREATE TABLE Series (
 SeriesID INT NOT NULL AUTO_INCREMENT,
 SeriesName VARCHAR(50) NOT NULL,
 PRIMARY KEY (SeriesID)
);

CREATE TABLE Venue (
 VenueID INT NOT NULL AUTO_INCREMENT,
 VenueStadium VARCHAR(50) NOT NULL,
 VenueCity VARCHAR(50) NOT NULL,
 VenueCountry VARCHAR(50) NOT NULL,
 PRIMARY KEY (VenueID)
);

CREATE TABLE `Match` (
 MatchID INT NOT NULL AUTO_INCREMENT,
 RefereeID INT NOT NULL,
 UmpireID INT NOT NULL,
 SeriesID INT NOT NULL,
 VenueID INT NOT NULL,
 Team1ID INT NOT NULL,
 Team2ID INT NOT NULL,
 TossWinnerTeamID INT NOT NULL,
 MatchWinnerTeamID INT NOT NULL,
 TossChoice VARCHAR(50),
 MatchResult VARCHAR(50),
 ManOfTheMatch VARCHAR(50),
 DebutPlayers VARCHAR(50),
 MatchName VARCHAR(50),
 MatchNo INT NOT NULL,
 MatchDate DATE,
 MatchFormat VARCHAR(50),
 PRIMARY KEY (MatchID),
 FOREIGN KEY (RefereeID) REFERENCES Referee(RefereeID),
 FOREIGN KEY (UmpireID) REFERENCES Umpire(UmpireID),
 FOREIGN KEY (SeriesID) REFERENCES Series(SeriesID),
 FOREIGN KEY (VenueID) REFERENCES Venue(VenueID),
 FOREIGN KEY (Team1ID) REFERENCES Team(TeamID),
 FOREIGN KEY (Team2ID) REFERENCES Team(TeamID),
 FOREIGN KEY (TossWinnerTeamID) REFERENCES Team(TeamID),
 FOREIGN KEY (MatchWinnerTeamID) REFERENCES Team(TeamID)
);

CREATE TABLE TeamCaptain (
 CaptainID INT NOT NULL AUTO_INCREMENT,
 MatchID INT NOT NULL,
 PlayerID INT NOT NULL,
 TeamID INT NOT NULL,
 PRIMARY KEY (CaptainID),
 FOREIGN KEY (MatchID) REFERENCES `Match`(MatchID),
 FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
 FOREIGN KEY (TeamID) REFERENCES Team(TeamID)
)
```

This script reflects the careful thought put into ensuring each table serves its unique function while seamlessly integrating with the entire database ecosystem.

SQL Refinement and Database Expansion in pgAdmin

In the progression from our initial database setup, I refined our SQL queries to be fully compatible with the PostgreSQL environment using pgAdmin's advanced tools. This refinement was critical in transitioning from MySQL conventions to PostgreSQL standards, ensuring smooth database operations and integrity.

For instance, we adapted from MySQL's AUTO_INCREMENT to PostgreSQL's SERIAL data type, a necessary change to maintain the auto-increment functionality for our primary keys. Here is an example showing the adjustment:

Old vs. New Query Illustration:

● **Sample Old Query:**

```
CREATE TABLE MatchPlayers (
 MatchPlayersID INT NOT NULL AUTO_INCREMENT,
 MatchID INT NOT NULL,
 PlayerID INT NOT NULL,
 TeamID INT NOT NULL,
 PRIMARY KEY (MatchPlayersID),
 FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
 FOREIGN KEY (TeamID) REFERENCES Team(TeamID)
);
```

● **Sample New Query:**

```
CREATE TABLE MatchPlayers (
   MatchPlayersID SERIAL PRIMARY KEY,
   MatchID INT NOT NULL,
   PlayerID INT NOT NULL,
   TeamID INT NOT NULL,
   FOREIGN KEY (MatchID) REFERENCES Match(MatchID),
   FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
   FOREIGN KEY (TeamID) REFERENCES Team(TeamID)
);
```

Moreover, I added the **'PlayerStats'** table to our database to deepen our analytical capabilities. This new table aggregates critical performance metrics, offering a holistic view of player achievements and trends. The creation of this table was done through the following SQL command: GitHub Link

```
CREATE TABLE PlayerStats (
 PlayerStatsID SERIAL PRIMARY KEY,
 PlayerID INT NOT NULL,
 TotalRuns INT DEFAULT 0,
 HighestScore INT DEFAULT 0,
 TotalWickets INT DEFAULT 0,
 FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID)
);
```

These enhancements optimized our database for the PostgreSQL ecosystem and set the stage for richer data analysis. You can review the complete, updated SQL script and the newly introduced ‘PlayerStats’ table in the repository linked below.

Translating Design into Function: The SQL Script Implementation

Building on the foundational work of the database design, the next step was to translate the Entity-Relationship Diagram into a working SQL script. This script serves as the instructions for the database, detailing how to set up each table and how they are linked together. The technical strength turns our conceptual plans into a functional T20 cricket database capable of managing intricate match details and player statistics. Feel free to visit my GitHub page, where I have documented the entire process.

Database Creation via pgAdmin:

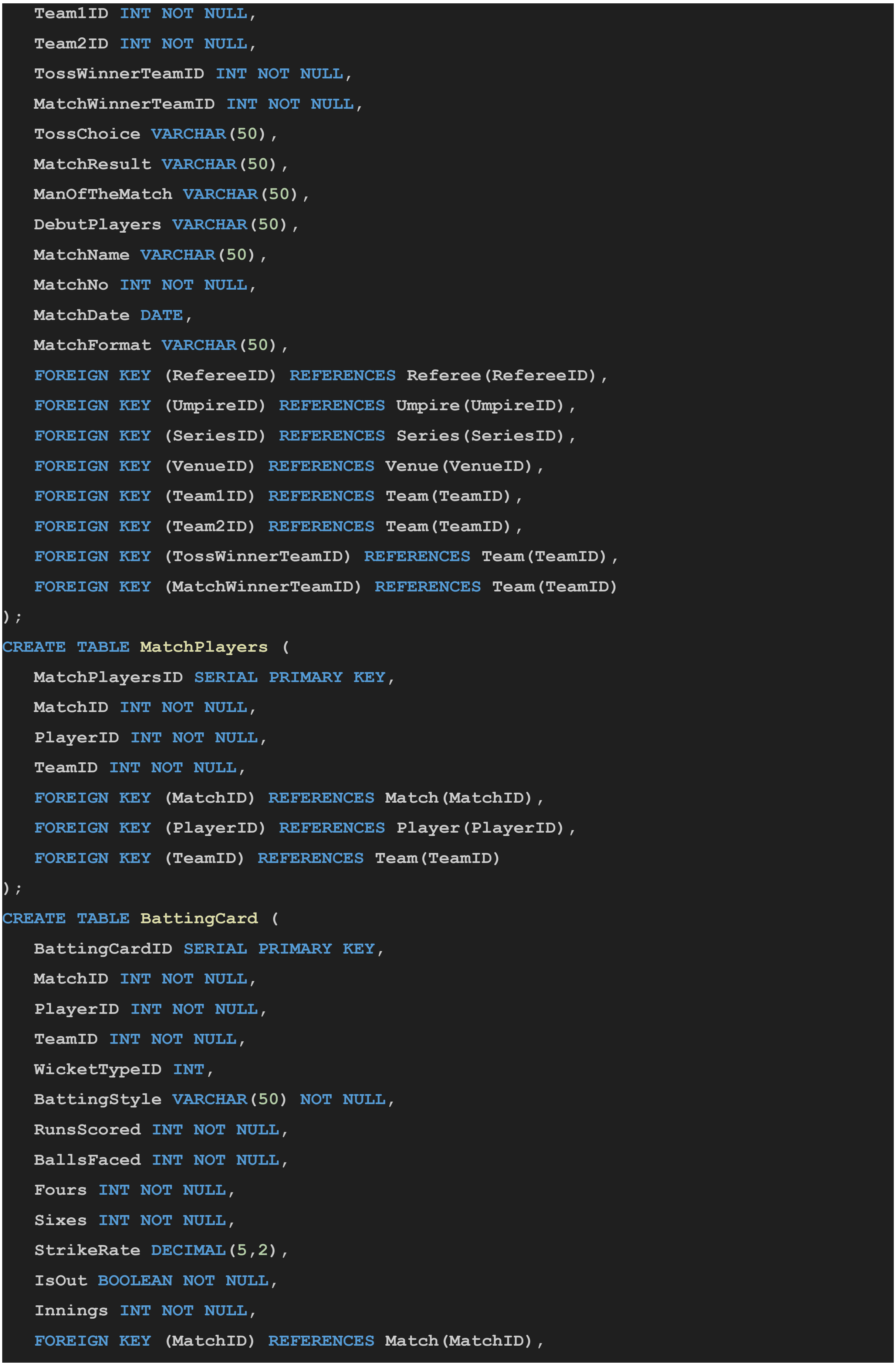
The database "All_T20_Internationals_Dataset_2005_2023" was constructed using pgAdmin's graphical interface, adhering to the following steps:

1. Initiated pgAdmin and connected to the PostgreSQL server.
2. Selected the 'Databases' node, right-clicked, and chose 'Create > Database'.
3. Provided the database name "All_T20_Internationals_Dataset_2005_2023" and confirmed the creation parameters.
4. Utilized the 'Query Tool' to execute the SQL script, thereby creating the schema's tables and relationships

SQL Script: Constructing the Database - GitHub

```sql
--CREATE DATABASE All_T20_Internationals_Dataset_2005_2023;
--USE All_T20_Internationals_Dataset_2005_2023;

CREATE TABLE Player (
   PlayerID SERIAL PRIMARY KEY,
   PlayerName VARCHAR(50) NOT NULL
);
CREATE TABLE Team (
   TeamID SERIAL PRIMARY KEY,
   TeamName VARCHAR(50) NOT NULL
);
CREATE TABLE Wicket (
   WicketTypeID SERIAL PRIMARY KEY,
   Bowled INT NOT NULL,
   Caught INT NOT NULL,
   Runout INT NOT NULL,
   LBW INT NOT NULL,
   Other INT NOT NULL
);
CREATE TABLE Referee (
   RefereeID SERIAL PRIMARY KEY,
   RefereeName VARCHAR(50) NOT NULL
);
CREATE TABLE Umpire (
   UmpireID SERIAL PRIMARY KEY,
   UmpireName VARCHAR(50) NOT NULL
);
CREATE TABLE Series (
   SeriesID SERIAL PRIMARY KEY,
   SeriesName VARCHAR(50) NOT NULL
);
CREATE TABLE Venue (
   VenueID SERIAL PRIMARY KEY,
   VenueStadium VARCHAR(50) NOT NULL,
   VenueCity VARCHAR(50) NOT NULL,
   VenueCountry VARCHAR(50) NOT NULL
);
CREATE TABLE Match (
   MatchID SERIAL PRIMARY KEY,
   RefereeID INT NOT NULL,
   UmpireID INT NOT NULL,
   SeriesID INT NOT NULL,
   VenueID INT NOT NULL,
```



```sql
   Team1ID INT NOT NULL,
   Team2ID INT NOT NULL,
   TossWinnerTeamID INT NOT NULL,
   MatchWinnerTeamID INT NOT NULL,
   TossChoice VARCHAR(50),
   MatchResult VARCHAR(50),
   ManOfTheMatch VARCHAR(50),
   DebutPlayers VARCHAR(50),
   MatchName VARCHAR(50),
   MatchNo INT NOT NULL,
   MatchDate DATE,
   MatchFormat VARCHAR(50),
   FOREIGN KEY (RefereeID) REFERENCES Referee(RefereeID),
   FOREIGN KEY (UmpireID) REFERENCES Umpire(UmpireID),
   FOREIGN KEY (SeriesID) REFERENCES Series(SeriesID),
   FOREIGN KEY (VenueID) REFERENCES Venue(VenueID),
   FOREIGN KEY (Team1ID) REFERENCES Team(TeamID),
   FOREIGN KEY (Team2ID) REFERENCES Team(TeamID),
   FOREIGN KEY (TossWinnerTeamID) REFERENCES Team(TeamID),
   FOREIGN KEY (MatchWinnerTeamID) REFERENCES Team(TeamID)
);
CREATE TABLE MatchPlayers (
   MatchPlayersID SERIAL PRIMARY KEY,
   MatchID INT NOT NULL,
   PlayerID INT NOT NULL,
   TeamID INT NOT NULL,
   FOREIGN KEY (MatchID) REFERENCES Match(MatchID),
   FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
   FOREIGN KEY (TeamID) REFERENCES Team(TeamID)
);
CREATE TABLE BattingCard (
   BattingCardID SERIAL PRIMARY KEY,
   MatchID INT NOT NULL,
   PlayerID INT NOT NULL,
   TeamID INT NOT NULL,
   WicketTypeID INT,
   BattingStyle VARCHAR(50) NOT NULL,
   RunsScored INT NOT NULL,
   BallsFaced INT NOT NULL,
   Fours INT NOT NULL,
   Sixes INT NOT NULL,
   StrikeRate DECIMAL(5,2),
   IsOut BOOLEAN NOT NULL,
   Innings INT NOT NULL,
   FOREIGN KEY (MatchID) REFERENCES Match(MatchID),
   FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
   FOREIGN KEY (TeamID) REFERENCES Team(TeamID),
   FOREIGN KEY (WicketTypeID) REFERENCES Wicket(WicketTypeID)
);
CREATE TABLE BowlingCard (
   BowlingCardID SERIAL PRIMARY KEY,
   PlayerID INT NOT NULL,
   MatchID INT NOT NULL,
   TeamID INT NOT NULL,
   BowlingOppositionTeamID INT NOT NULL,
   Innings SMALLINT NOT NULL,
   OversBalled SMALLINT NOT NULL,
   TotalBalls SMALLINT NOT NULL,
   Maidens SMALLINT NOT NULL,
   RunsConceded SMALLINT NOT NULL,
   WicketsTaken SMALLINT NOT NULL,
   EconomyRate DECIMAL(4,2) NOT NULL,
   DotBalls SMALLINT NOT NULL,
   FoursGiven SMALLINT NOT NULL,
   SixesGiven SMALLINT NOT NULL,
   WidesBowled SMALLINT NOT NULL,
   NoBalls SMALLINT NOT NULL,
   FOREIGN KEY (MatchID) REFERENCES Match(MatchID),
   FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
   FOREIGN KEY (TeamID) REFERENCES Team(TeamID),
   FOREIGN KEY (BowlingOppositionTeamID) REFERENCES Team(TeamID)
);


CREATE TABLE TeamCaptain (
   CaptainID SERIAL PRIMARY KEY,
   MatchID INT NOT NULL,
   PlayerID INT NOT NULL,
   TeamID INT NOT NULL,
   FOREIGN KEY (MatchID) REFERENCES Match(MatchID),
   FOREIGN KEY (PlayerID) REFERENCES Player(PlayerID),
   FOREIGN KEY (TeamID) REFERENCES Team(TeamID)
)
```

Pros and Cons: Non-Relational Database Implications for T20 Cricket Data - Link

Pros and Their Implications:

1. **Enhanced Flexibility**:
    - Implication: Easily incorporate diverse data like match photos, player interviews, or social media interactions without schema restructuring.
    - Example: Seamlessly add a new column for player social media handles or match highlight videos.
2. **Improved Scalability**:
    - Implication: Effortlessly handle increasing data during high-profile T20 tournaments or as the sport gains global popularity.
    - Example: Rapidly scale up storage during the World Cup season to accommodate the influx of match data and fan interactions.
3. **Speedy Data Retrieval**:
    - Implication: Quicker access to player statistics and match data, enhancing real-time analytics during live games.
    - Example: Instantly pull up a player's performance history during a live match for on-the-spot analysis.
4. **Cost-Effectiveness**:
    - Implication: Reduce overheads linked to data storage and maintenance, which is especially beneficial for extensive, evolving datasets.
    - Example: Lower costs associated with storing years of historical match data and ongoing season statistics.
5. **Real-time Processing**:
    - Implication: Enable immediate processing and analysis of data as matches unfold, which is crucial for live updates and analytics.

  - Example: Real-time analysis of bowling patterns and batting strategies during a live broadcast.

6. **Developer-Friendly**:
   - Implication: It is easier for developers to interact with the database, speeding up the development of new analytics tools.
   - Example: Quickly develop and deploy a new app feature that tracks player fitness levels throughout the season.

Cons and Their Implications:

1. **Consistency Concerns**:
   - Implication: There is potential for slight discrepancies in real-time data, impacting the accuracy of in-the-moment analytics.
   - Example: A slight delay in updating a player's run total during a live match.
2. **Transactional Support**:
   - Implication: Complex transactions, like batch updates to player statistics, might be more cumbersome to manage.
   - Example: Difficulty in executing simultaneous updates to multiple player records after a match.
3. **Learning Curve**:
   - Implication: The team may initially face challenges adapting to new technologies, possibly slowing down data updates or analysis.
   - Example: Analysts require additional training to adapt to new query languages or database structures.
4. **Integration Issues**:
   - Implication: Challenges in integrating the new system with existing tools and workflows designed for relational databases.

- Example: Existing analytics tools may require significant adjustments or redevelopment to work with the new database.

5. **Query Performance**:
    - Implication: Complex queries may run slower, like those combining multiple data types or large datasets.
    - Example: A sophisticated query that combines player stats, historical data, and real-time social media feeds might take longer to execute.
6. **Support and Community**:
    - Implication: Finding solutions to specific problems or optimizing the database for unique use cases might be more challenging.
    - Example: Limited resources or community support for troubleshooting an unusual data retrieval issue specific to T20 data.

Recommendation on Transitioning to a Non-Relational Database - Link

1. **Agility for Analytics:**
    - Benefit: Enhanced capability for rapid querying across large datasets, essential for the in-depth analysis of T20 cricket data.
    - Implication: Enables quick insights from player performances to match outcomes, supporting real-time decision-making.
2. **Rich Media Management:**
    - Benefit: The ability to effortlessly handle a variety of data types, from numerical stats to multimedia like match images and videos.
    - Implication: Provides a holistic view of matches, enriching fan experiences and broadening analytical perspectives.

3. **Scalability for Growth:**
    - Benefit: Non-relational databases can scale horizontally, which is especially important during peak times like major tournaments.
    - Implication: Ensures the database performs optimally even under the pressure of increased data influx.
4. **Live Event - Data Capture:**
    - Benefit: Real-time data capture capabilities extend to live match details, including player tracking and audience reactions.
    - Implication: Offers an enriched dataset for dynamic analysis and enhanced viewer engagement strategies.
5. **Machine Learning Readiness:**
    - Benefit: A non-relational framework eases the integration and analysis of diverse datasets for machine learning applications.
    - Implication: Sets the stage for advanced predictive modeling and automated insights, crucial for forward-looking cricket analytics.
6. **Match Forecasting:**
    - Benefit: Facilitates more accurate and nuanced predictive analytics, giving more profound insights into potential match outcomes and trends.
    - Implication: Enhances strategic planning and offers a competitive edge in understanding the game's dynamics.

The transition to a non-relational database aligns with our goals to enhance T20 cricket analytics, offering a future-proof solution that caters to the growing, dynamic needs of the sport. Its flexibility, scalability, and capability to handle diverse data types make it an ideal choice for our evolving data requirements.

Deployment Plan for Upgrading to a Non-Relational Database

1. Pre-Planning Phase:

- **Assessment and Requirements Gathering:**
  - Evaluate the current database structure and determine the requirements for the non-relational system.
  - Identify critical data types, relationships, and performance needs specific to T20 cricket analytics.
- **Selecting the Right Non-Relational Database:**
  - Choose a database that best aligns with the needs of T20 cricket data (consider options like MongoDB or Cassandra.
- **Resource Allocation and Team Formation:**
  - Assemble a team of database experts, developers, and analysts.
  - Allocate necessary resources, including hardware and software requirements.

2. Execution Phase:

- **Data Migration Strategy:**
  - Develop a thorough data migration plan that guarantees data integrity and minimal downtime.
  - Implement data conversion scripts if necessary.
- **Database Configuration and Setup:**
  - Install and configure the selected non-relational database.
  - Define data models and schemas as per the new system's requirements.
- **Integration with Existing Systems:**
  - Ensure compatibility and integration with existing applications and analytics tools.

- **Testing:**
    - Perform thorough testing for data accuracy, performance, and reliability.

3. Post-Implementation Phase:

- **Training and Documentation:**
    - Provide training to the team on the new database system.
    - Update documentation to reflect new processes and structures.
- **Monitoring and Optimization:**
    - Monitor the system for performance and scalability.
    - Continuously optimize queries and storage for efficiency.
- **Feedback Loop and Incremental Improvements:**
    - Establish a feedback mechanism to gather insights on the new system's performance.
    - Implement incremental improvements based on feedback and evolving needs.

4. Timeline:

- Outline a realistic timeline for each phase, considering the complexity of the migration and team availability.

5. Risk Management:

- Identify potential risks and challenges during the migration process and develop contingency plans.

Final Overview:

- The deployment plan aims to ensure a smooth transition to a non-relational database, enabling more effective management and analysis of T20 cricket data with minimal disruption to current operations.

Overall Conclusion on the T20 Cricket Database Upgrade

As I bring this comprehensive project to a close, I reflect on the journey of upgrading our T20 cricket database to a non-relational system. Creating the data dictionary and the Entity-Relationship Diagram (ERD) was pivotal in laying a clear foundation for our database's structure. The SQL code I crafted represents the technical bedrock upon which our database functions efficiently.

In assessing the pros and cons of making this transition, I have balanced the potential benefits against the challenges, ensuring a well-informed decision-making process. My recommendation to move towards a non-relational database stems from a thorough analysis of its adaptability, performance benefits, and suitability for our diverse and dynamic cricket data.

The deployment plan I developed details a structured approach to this upgrade, prioritizing careful planning, strategic execution, and ongoing refinement. This plan is a testament to my commitment to enhancing our data management capabilities while ensuring a seamless transition.

This upgrade is more than just a technical shift; it is a strategic step in handling and analyzing cricket data. It opens up new possibilities for more sophisticated, real-time analytics, enriching our insights into the captivating world of T20 cricket.

References:

MySQL.(n.d.). 9.3 *Keywords and Reserved Words*. https://dev.mysql.com/doc/refman/8.0/en/keywords.html#:~:text=Certain%20keywords%2C%20such%20as%20SELECT,names%20of%20built%2Din%20functions.&text=Names%20of%20built%2Din%20functions%20are%20permitted%20as%20identifiers%20but,to%20be%20used%20as%20such.

PRASAD, B. (2023, October 23). *All T20 Internationals Dataset (2005 - 2023)*. Kaggle. https://www.kaggle.com/datasets/bhuvaneshprasad/all-t20-internationals-dataset-2005-to-2023?select=t20i_Matches_Data.csv

Pgadmin.(n.d.). *Login/Group Role Dialog.* https://www.pgadmin.org/docs/pgadmin4/development/role_dialog.html

Pgadmin.(n.d.). *ERD Tool.* https://www.pgadmin.org/docs/pgadmin4/development/erd_tool.html

Siggard, R., Dupin-Bryant, P. A., Mills, R. J., & Olsen, D. H. (2022). *Teaching Tip: Using SQL to Create and Mine Large, Customizable Datasets.* Journal of Information Systems Education, https://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=bsu&AN=158589553&Custid=083-900

Taylor, A. G. (2019, March 31). *How to use the SQL union join.* Dummies. https://www.dummies.com/article/technology/programming-web-design/sql/how-to-use-the-s

ql-union-join-260866/