

## **Abstract**

Cricket in India is more than merely a game; it is a matter of deep concern for the people and an activity that brings people from different backgrounds together. T20 Internationals (T20Is) are quick and dynamic, and the usual stats (runs, strike rate, wickets) fail to capture the full depth of a player's performance. Therefore, our study has a data-driven, context-aware approach to T20I cricket analysis with a specific type of graph database known as Neo4j which is perfect for representing intricate and interrelated relationships. Players, matches, performances, innings and situations are represented as nodes and the connections between them are relationships. Relational databases don't work well when such highly interconnected data is there but graph databases like Neo4j are designed for this. They allow us to comfortably analyze things such as player partnerships, which pitch a player bats in, who they're playing against and pitch types. Neo4j makes it particularly suitable to find hidden trends and interdependencies in cricket that are otherwise hard to spot. Rather than focusing on typical stats of a player, this method provides us with a holistic and inter-related picture of how the players play, how the teams play against each other and what actually makes a difference, which finally leads to better-informed and better-intelligent decision-making in T20I cricket.

## **Introduction**

Cricket, the enthralling game that stirs the emotions of millions, has expanded beyond its sports confines in India. It's more than simply a sport; it's a cultural phenomenon and a unifying force. Cricket also contributes significantly to the country's economy with the IPL generating lots of revenue and media coverage. The growing significance of data-driven insights in sports, particularly in cricket has created new opportunities for deeper performance analysis and strategic decision-making.

The T20 matches are much shorter than other types of matches like ODI and Test Matches, so every player has to give his best, there is no such time to settle in. The T20I matches generate large volumes of data that include not just runs or wickets but also certain details like who played with whom, where the match happened, what were the pitch conditions, etc. The older ways of measuring performance of players, just by looking at the averages and total runs don't tell the whole story, they miss the details that really affect how a cricketer plays. The factors such as batting positions, partnerships, opposition team, pitch conditions and match pressure affects player's performance and overall impact. Hence, if we only look at the basic statistics it may lead to misunderstanding of how well someone is actually playing and result in oversimplified interpretations.

The main idea of our study was to look at cricket data in a smarter way and model and analyze the performance in a more connected and context aware manner. Therefore, rather than considering each stat on its own, we are observing the interrelationships between players, matches, roles and conditions to understand how they interact. Such an integrated view can help

find deeper insights, make more well-informed data-driven decisions, strategic planning and accurate performance evaluation in T20I cricket.

Data modeling frameworks can be classified at large into two categories - relational and non-relational. Traditional relational databases ( like MySQL, Oracle or PostgreSQL) store data in rows and columns with fixed schema. They perform efficiently when data is well structured and the relationship between data points is not very complex. But in T20I cricket, the relationships are much more intricate, one player can play a lot of matches, play different roles, play with different players, play under various conditions and play against different countries. Employing a relational database in this would mean having multiple tables and complicated joins, thus it would be slow, hard to maintain and also hard to query. This is where NoSQL (Not Only SQL) databases are clearly beneficial. NoSQL databases can be classified into four main types : Document-based databases (e.g., MongoDB, CouchDb), Key-value stores (e.g., Redis, Amazon DynamoDB), Column-oriented databases (e.g., Apache Cassandra, Google Bigtable) and Graph-based databases (e.g., Neo4j, TigerGraph, OrientDB) based on their data storage and retrieval methods. NoSQL databases do not require rigid structure, it is made to be more adaptable. Among NoSQL data models, graph databases like Neo4j are best applied to rich, many-to-many interconnectivity datasets, for example, cricket where players, matches, innings and conditions all have high interlinkages. Rather than foreign keys and joining tables, graph databases are based on a node and relationship structure that is more convenient and efficient to follow links.

**Cricket data is full of connections** - players, matches, innings, and performance contexts that all interact in dynamic ways, which makes Neo4j a great fit for analysis. In Neo4j, everything like players, matches, and situations is stored as a “node,” with relationships linking them, such as a player facing a specific opponent or playing in a particular match. This setup allows us to easily explore how different factors impact player performance, like how a batsman performs under pressure or which combinations work best.\*\*\*\*\* Unlike traditional databases, Neo4j makes it easy to analyze these complex connections. We chose Neo4j because it’s user-friendly, has a simple query language (Cypher), and offers excellent performance. It helps us build an insightful cricket analytics system that uncovers hidden patterns and provides meaningful analysis of player performance.

## Objective

The goal of this research is to analyze T20 player performance and find out who does best in different batting positions, on various fields, against specific opponents, and in match situations like powerplay, middle over, or death over using graph database framework (Neo4j)

**The tasks that will be done to implement this objective are:**

1. **Nodes like player, team, match, field, innings, batting position, match situation, and opponent will be connected through relationships to create a connected picture. Cricket-related details**

such as match conditions, batsman-bowler roles, and opponents are added to the graph.

2. Analysis is done using Cypher queries to find who played well in certain conditions, which batting position suited whom, who performed better on specific grounds, or which batsman scored more against which bowler.

## Contribution

This research offers a new approach to T20 cricket analysis using the Neo4j graph database, focusing on relationships rather than just basic stats like runs and averages. We analyze connections between players, matches, grounds, opponents, batting positions, and innings.

The main contributions of this study are:

1. **\*\*\*\*\* Dataset preprocessing**
2. **Building Graph Based Cricket Knowledge Base:** Neo4j helps create cricket Knowledge Base and allows easy addition of new metrics or relationships, such as tracking players' performance in death overs or strong partnerships.
3. **Graph analysis of player's performance evaluation:** This research goes beyond basic statistics by analyzing player performances and their relationships, like how a batsman performs against a particular bowler or at a specific time.
4. **Complex Analysis of multi-hop relationships using Cypher Queries:** Neo4j makes it easy to analyze complex relationships, such as checking performance of a partnership during a specific phase under certain conditions. Also, finding matches where a player chasing runs, played on a batting-friendly pitch, and performed better than average can be done in one Cypher query. Traditional databases require multiple join operations (increasing cost) to solve such queries.

## Proposed Methodology

The proposed framework uses Neo4j as a graph database to develop a system for performing cricket player performance analysis through multidimensional relational patterns. Representing the data in a graph structure allows us to capture the diverse relationships among cricket concepts including players and teams as well as matches and performance information.

To build this, we curated a rich dataset[link] by integrating information from multiple reliable sources, to capture the overall and situational performances. From ESPNcricinfo<sup>[1]</sup>, we collected data up to 2024, including players' overall stats, position wise records, performance based on opponent, spin vs pace performance, home and away statistics across three areas-batting, bowling and fielding. We also used ball-by-ball data from Cricsheet<sup>[2]</sup>, originally in JSON format later converted to CSV from to extract specific match information for matches played between July 2022 and July 2024 which covered players' performance in every match and different phases like the powerplay, middle over and death overs and calculated some key metrics such as match wise batter strike rates, bowler economies and other important statistics. We also included ground wise, pitch specific data from HowSTAT<sup>[3]</sup>. Team rankings sourced from ICC Men's team rankings<sup>[4]</sup> to calculate players performance against different teams. Based on this dataset[link],



## Key Nodes:

- **Players:** Represents individual cricket players.
  - **Match:** Represents individual match details.
  - **Innings:** Represents details of the innings.
  - **Team:** Represents a team participating in the match
  - **Ground:** Locations where matches are hosted.
  - **BattingPosition:** Batting performance by position.
  - **DismissalStats:** Records how batters were dismissed.
  - **Partnership:** Statistics of partnerships formed between two batters.
  - **SpinPaceStats:** Player performance against spin vs pace bowling.
- 
- **Against Opponent**
    - **BattingAgainstOpponentStats:** Player's batting performance against specific opponents.
    - **BowlingAgainstOpponentStats:** Player's bowling performance against specific opponents.
  - **Home/Away**
    - **BattingAwayStats:** Batting statistics for matches played away from home.
    - **BowlingAwayStats:** Bowling statistics for matches played away from home.
    - **BattingHomeStats:** Batting statistics for matches played at home grounds.
    - **BowlingHomeStats:** Bowling statistics for matches played at home grounds.
  - **Overall Stats**
    - **OverallBattingPerformance:** Aggregated overall batting statistics across all matches.
    - **OverallBowlingPerformance:** Aggregated bowling stats across all matches.
    - **OverallFieldingPerformance:** Overall fielding performance across all matches.
  - **Matchwise Stats**
    - **MatchWiseBattingPerformance:** Batting stats broken down match by match.
    - **MatchWiseBowlingPerformance:** Bowling stats for each match individually.
    - **MatchWiseFieldingPerformance:** Fielding stats broken down by individual matches.
    - **MatchWiseWicketkeepingPerformance:** Wicketkeeping performance stats for each match.
  - **Clutch Performance**
    - **ClutchBattingPerformance:** Batting stats in high-pressure or clutch match situations.
    - **ClutchBowlingPerformance:** Bowling performance in clutch situations.

- **ClutchFieldingPerformance**: Fielding performance in clutch or pressure situations.

*\*Clutch Performance: Performance of individual players in key phases like powerplay, middle overs or death overs against specific opponents.*

### Key Relationships:

- **[BATSMAN\_IN]**: Links players to their batting performance in a specific match.
- **[BOWLER\_IN]**: Links players to their bowling performance in a specific match.
- **[FIELDING\_IN]**: Links players to their fielding performance in a specific match.
- **[AGAINST\_TEAM]**: Links players to their performance while facing a specific team.
- **[BATTING\_AWAY] / [BATTING\_HOME]**: Links players to their Away/Home batting performance.
- **[CLUTCH\_BATTING]**: Links players to their clutch batting performance.
- **[CLUTCH\_BOWLING]**: Links players to their clutch bowling performance.
- **[HOSTED\_IN]**: Links the match to the ground where the match was hosted.
- **[IN\_MATCH]**: Links players to their participation in a match.

Traditional relational models struggle with multi-hop queries, context relationships, and dynamic exploration — all of which can yield rich insights towards Cricket performance analysis. And, to manage that, our framework suggests five principal components which are explained along with example queries and their results are shown :

### 1. Relational Depth Beyond Tables

Our graph model outperforms flat tables in RDBMS since it shows deep and interconnected relationships that identify players' match situation effects. The system uses specific and meaningful relationships like **[BATSMAN\_IN]** and **[CLUTCH\_BATTING]** to address complex queries.

### Example Queries:

#### 1.1 Match Influencers (All-Rounders in a Single Match)

```
MATCH (p:Players)-[:BATSMAN_IN]->(bat)-[:IN_MATCH]->(m:Match),
      (p)-[:BOWLER_IN]->(bowl)-[:IN_MATCH]->(m),
      (p)-[:FIELDING_IN]->(field)-[:IN_MATCH]->(m)
WHERE bat.Runs > 20 AND bowl.wickets > 1 AND field.Catches > 0
RETURN p.Name AS 'Player Name', m.date AS 'Match Date', bat.Runs AS 'Runs', bowl.wickets AS Wickets,
field.Catches AS Catches
```

#### Output 1.1:

	Player Name	Match Date	Runs	Wickets	Catches
1	"Axar Patel"	2023-01-05	65	2	1
2	"Hardik Pandya"	2022-07-07	51	4	1
3	"Hardik Pandya"	2022-10-23	40	3	1
4	"Hardik Pandya"	2023-02-01	30	4	2
5	"Shivam Dubey"	2024-07-14	26	2	1
6	"Washington Sundar"	2023-01-27	50	2	1

## 1.2 Overall and Phase wise batting performance Against Australia

MATCH

(p:Players)-[:BAT\_AGAINST]->(ba:BattingAgainstOpponentTeam)-[:AGAINST\_TEAM]->(:Team {name: "Australia"}),

(p)-[:CLUTCH\_BATTING]->(cb:ClutchBattingPerformance)

WHERE ba.Runs IS NOT NULL AND cb.Opponent = "Australia"

RETURN p.Name AS Player, ba.Runs AS `Total Runs Vs Australia`, cb.Innings AS Innings, cb.PowerplayTotal AS `Powerplay Runs`,

cb.MiddleOversTotal AS `MiddleOvers Runs`, cb.DeathOversTotal AS `DeathOvers Runs`, cb.DeathOversSixes AS `Death Overs Sixes`

ORDER BY `Total Runs Vs Australia` DESC

### Output 1.2:

	Player	Total Runs Vs Australia	Innings	Powerplay Runs	MiddleOvers Runs	DeathOvers Runs	Death Overs Sixes
1	"Virat Kohli"	794	4	38	25	13	1
2	"Rohit Sharma"	484	4	112	33	0	0
3	"Suryakumar Yadav"	290	9	79	181	30	2
4	"KL Rahul"	271	3	25	30	0	0
5	"Hardik Pandya"	235	4	9	30	93	7
6	"Ruturaj Gaikwad"	223	4	36	107	67	7
7	"Yashasvi Jaiswal"	138	5	118	0	0	0
8	"Ishan Kishan"	110	3	17	92	0	0
9	"Rinku Singh"	105	4	0	45	60	2
10	"Shreyas Iyer"	73	2	9	33	19	1

## 2. Exploratory & Flexible Querying

The Neo4j graph database enables ad hoc traversal of paths and contextual relationships, which allows for exploratory research and dynamic querying. The database allows experts to observe player performance details within certain match scenarios and conditions.

### Example Query:

#### Partnership Impact with Tilak Verma & Death Over Potential

```
MATCH (pl:Players {Name: "Tilak Verma"})-[:FORMED]->(p:Partnership)-[:FORMED]-(partner:Players)
MATCH (partner)-[:CLUTCH_BATTING]->(cb:ClutchBattingPerformance)-[:AGAINST_TEAM]->(Team {name:
"Australia"})
MATCH (partner)-[:BATTING_AWAY]->(ba:BattingAwayStats)
WHERE ba.Average > 40 AND cb.DeathOversTotal >= 30
RETURN DISTINCT partner.Name AS PlayerName, p.RunsDescending AS `Partnership Runs`, p.High AS
`Highest Partnership`,
       ba.Average AS `Away Average`, cb.PowerplayTotal AS `Powerplay Runs`, cb.MiddleOversTotal AS
`Middle Overs Runs`,
       cb.DeathOversTotal AS `Death Overs Runs`, cb.DeathOversSixes AS `Death Overs Sixes`
ORDER BY cb.DeathOversTotal DESC
```

### Output 2:

	PlayerName	Partnership Runs	Highest Partnership :	Away Average	Powerplay Runs	Middle Overs Runs	Death Overs Runs	Death Overs Sixes
1	"Ruturaj Gaikwad"	243	141	49.0	36	107	67	7
2	"Suryakumar Yadav"	313	87	48.55	79	181	30	2

## 3. Performance on Multi-Hop Queries

Deep joins in relational databases cause performance problems during nested and indirect relationship queries. The graph model efficiently processes multiple hop queries.

### Example Query:

#### Bowling Performance Against Top Teams

```
MATCH(bowler:Players)-[:CLUTCH_BOWLING]->(cb:ClutchBowlingPerformance)-[:AGAINST_TEAM]->(tea
m:Team)
WHERE team.team_rank <= 4 AND team.name <> "India" AND cb.DeathOversBalls > 0
WITH bowler, team, toFloat(cb.DeathOversRuns) / (cb.DeathOversBalls / 6.0) AS DeathOverEconomy
WHERE DeathOverEconomy < 10.0
MATCH (bowler)-[:BOWLING_AWAY]->(ba:BowlingAwayStats)
WHERE ba.Wickets > 10
RETURN DISTINCT bowler.Name AS Bowler, team.name AS `Opponent Team`, ba.Wickets AS `Away Wickets`,
```



round(DeathOverEconomy, 2) AS `Economy in Death Vs Team`  
ORDER BY `Away Wickets` DESC, `Economy in Death Vs Team` ASC

### Output 3:

	Bowler ≡	Opponent Team	Away Wickets	Economy in Death
1	"Hardik Pandya"	"Australia"	34	7.5
2	"Kuldeep Yadav"	"New Zealand"	33	5.0
3	"Jasprit Bumrah"	"England"	31	4.5
4	"Jasprit Bumrah"	"Australia"	31	9.75
5	"Arshdeep Singh"	"England"	24	4.67
6	"Arshdeep Singh"	"New Zealand"	24	9.88
7	"Ravi Bishnoi"	"England"	21	2.0
8	"Washington Sundar"	"New Zealand"	18	6.0
9	"Mukesh Kumar"	"Australia"	13	8.33

## 4. Visual Intelligence Layer: Cricket Knowledge Base

We can also view the cricket knowledge graph in an intuitive format. Nodes represent players, matches, grounds, performance, innings, and edges represent relationships amongst the nodes.

### Visual Insights Supported:

- **Node connection pathways:** We can show different pathways amongst nodes.  
**For example-** Player → Bowler → Match → Ground
- **Performance Trajectories:** We can track performance of a player over time.
- **Relationship-based Networks:** We can form networks to show various connections between players.

### Example Query:

#### Bowling Performance Chain:

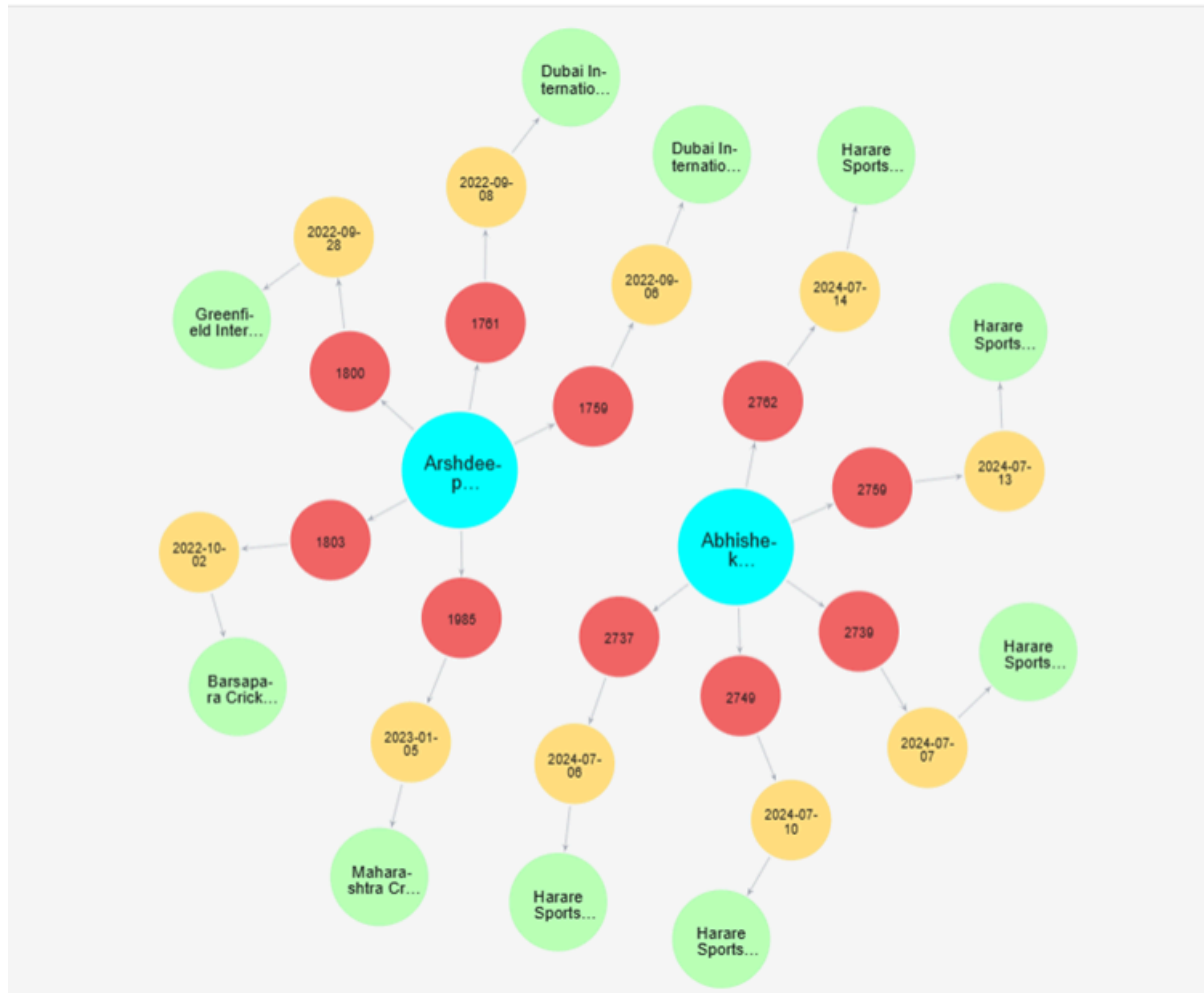
```
MATCH (player:Players)-[r1:BOWLER_IN]->(bs:MatchWiseBowlingPerformance)
-[r2:IN_MATCH]->(match:Match)
```

```

-[r3:HOSTED_IN]->(ground:Ground)
RETURN player, r1, bs, r2, match, r3, ground
LIMIT 10

```

#### Output 4:



### 5. Schema Flexibility for New Metrics

Unlike traditional RDBMS architectures—where adding new metrics like phase-wise strike rates often demands schema alterations, new columns, or complex JOINS—our graph-based approach in Neo4j enables seamless, non-disruptive extensions.

#### Example Query:

##### Adding Phase-wise Strike Rate in Clutch Batting Performance Node.

```

MATCH (cbp:ClutchBattingPerformance)
WITH cbp,
CASE WHEN cbp.PowerplayBallsFaced > 0
THEN round(toFloat(cbp.PowerplayTotal) / cbp.PowerplayBallsFaced * 100, 2)


```

```

ELSE 0 END AS powerplaySR,
CASE WHEN cbp.MiddleOversBallsFaced > 0
THEN round(toFloat(cbp.MiddleOversTotal) / cbp.MiddleOversBallsFaced * 100, 2)
ELSE 0 END AS middleSR,
CASE WHEN cbp.DeathOversBallsFaced > 0
THEN round(toFloat(cbp.DeathOversTotal) / cbp.DeathOversBallsFaced * 100, 2)
ELSE 0 END AS deathSR
SET cbp.powerplayStrikeRate = powerplaySR,
    cbp.middleOversStrikeRate = middleSR,
    cbp.deathOversStrikeRate = deathSR
RETURN cbp.Name AS Player, cbp.Opponent AS Opponent,
cbp.powerplayStrikeRate AS PowerplaySR, cbp.middleOversStrikeRate AS MiddleSR, cbp.deathOversStrikeRate
AS DeathSR
ORDER BY MiddleSR DESC

```

### Output 5:

	Player	Opponent 	PowerplaySR	MiddleSR	DeathSR
1	"Suryakumar Yadav"	"Hong Kong"	0	371.43	221.05
2	"Ravindra Jadeja"	"Sri Lanka"	0	323.53	78.95
3	"Shivam Dubey"	"South Africa"	0	300.0	109.09
4	"Suryakumar Yadav"	"Netherlands"	0	288.89	156.25
5	"Rinku Singh"	"Zimbabwe"	0.0	250.0	178.57
6	"KL Rahul"	"Bangladesh"	115.79	245.45	0
7	"Axar Patel"	"Sri Lanka"	0	234.62	169.23

## Conclusion

The progress of data modelling alongside graph-based technologies enables us to create a framework that provides superior analytics of sport player performances particularly in cricket. Our system handles each aspect of analytical depth, flexible information querying, multi-node traversal capabilities, and visualization outputs independently. This framework will assist us in demystifying how different factors such as pitch type, strength of the opponent, role of the player, batting position, etc., impacts the performance of a player. We can also examine the performance of different players playing in different types of conditions-at home and outside home, against spinners and against pacers etc. In total, this model provides the basis for a deeper penetration of player performances in the game of cricket, allowing scalable and context-sensitive insights that are usually lost with conventional models as a result of several complexities.

## Future Work

In the future, the insights from this research can be visualized through Neo4j Bloom dashboards, making it easier for analysts to interpret the data. We also aim to explore innovative concepts in team formation and player performance prediction, which will help coaches and selectors predict when, where, and how players are likely to perform at their best. This will ultimately support more strategic decision-making in team selection and game planning.

## References

1. <https://stats.espncricinfo.com/ci/engine/stats/index.html>
2. <https://cricsheet.org/matches/>
3. <https://www.howstat.com/Cricket/Home.asp>
4. <https://www.icc-cricket.com/rankings>
5. <https://www.geeksforgeeks.org/types-of-nosql-databases/>
6. <https://www.restack.io/p/neo4j-answer-graph-databases-comparison-cat-ai>
7. <https://www.geeksforgeeks.org/difference-between-sql-and-nosql/>
8. <https://culturalindia.org.in/cricket-the-cultural-and-religious-sport-of-india/>
9. <https://www.geeksforgeeks.org/neo4j-introduction/>