# IPL

Chinmaya Singh (2016EE30220), Adarsh Agarwal (2016CE10213), Vivek Singal (2016EE10421)

Due date: February 26, 2019, 11:55pm IST

## 1 Project Description

Our project has everything a cricket enthusiast wants to know about IPL. We have the data of every ball played in IPL from season 2008 to 2017. We have written many sql queries so that our user can see all sorts of statistics possible in cricket. User can search a player, see who have the most runs, wickets, strikerate etc. They can search a match and see its scorecard.

The website is self-sustainable. We have given some users the permission to add matches that will be played in the future. We have a robust database design with appropriate constraints and triggers to prevent users from adding wrong data.

## 2 Data Sources and Statistics

We downloaded our dataset from https://www.kaggle.com/raghu07/ipl-data-till-2017.
The data was available in .csv format. So we direclty downloaded it and analyzed it using MS Excel.
The date was structured but it was not clean.

- Most of the data cleaning was done in MS Excel itself. For eg. I replaced 'Legbyes' with 'legbyes' in ball(Extra_type) and many more such anomalies such that the letter cases are consistent through out the database.
- We have removed many columns from the raw data set which were ending with 'sk'. Such columns were actually surrogate keys generated by the earlier database application.
- We have also removed many redundant columns giving information which can be actually extracted out from combination of other multiple columns or tables.
- We have tried our best to normalize our tables. In the process of normalization, we had to make two more tables venue and team_match.

| **Entity** | **Attributes** |
|---|---|
| team | team_id, team_name |
| player | player_id, player_name, dob, batting_hand, bowling_skill, country_name |
| venue | venue_id, venue_name, city_name, country_name |
| match | match_id, team1, team2, match_date, season_year, venue_id, toss_winner, match_winner, toss_name, win_type, outcome_type, manofmatch, win_margin |
| player_match | match_id, player_id, role_desc, player_team |
| team_match | match_id, innings_no, team_batting, team_bowling |
| ball | match_id, over_id, ball_id, innings_no, striker_batting_position, extra_type, runs_scored, extra_runs, out_type, striker, non_striker, bowler, player_out, fielder |
| login | uname, password |

Table 1: List of Entities and Attributes

| Table | No. of Tuples | Raw Dataset Size | Cleaned Dataset Size | Dataset size in DB | Time to Load |
|---|---|---|---|---|---|
| team | 13 | 343B | 306B | 8KB | 7.8200ms |
| player | 497 | 33.8KB | 32KB | 48KB | 10.008ms |
| venue | 34 | - | 1.59KB | 8KB | 7.234ms |
| match | 637 | 110.7KB | 83.1KB | 104KB | 21.351ms |
| player_match | 13992 | 2.54MB | 516KB | 928KB | 308.595ms |
| team_match | 1284 | - | 18KB | 80KB | 31.145ms |
| ball | 150451 | 23.94MB | 8.89MB | 15.39MB | 4733.687ms |
| login | 13 | - | - | 8KB | - |

Table 2: Data Statistics

# 3 Functionality and Working

## 1. User's View of the System

- **Home:**
  This is the main page of the website which does not conatin any SQL queries. It just contains some information about IPL.
  Disclaimer: The information is copy pasted from wikipedia
- **Teams:**
  The page just displays all the teams in table format which have played in IPL since its inception. It also includes the teams which were banned from playing or have not played in the last season or are no longer playing in the current season.
- **Players:**
  The page displays all the players in table format which have ever played in any IPL season since its inception.
  The webpage contains a filter which the user can use to select players with certain characteristics. The filter is based on the country of origin of the player, the hand which the player uses to bat and the hand which the player uses to bowl. The user can select multiple countries at a given time to select players from multiple countries.
  User can also search a player by name and get his statistics.
- **Matches:**
  This page displays all the matches in the list format. There is a filter below the menubar by which we can see the matches year wise. If the user uses the filter, then the page displayes the points table for that year along with the list of matches of that year.
  The title of each item of the list(i.e. match) is itself a hyperlink. If the user clicks on the hyperlink, he/she is redirected to a page which displays the scorecard of the match.
- **Match Scorecard:**
  The scorecard itself is divided into two webpages for Innings 1 and 2 of the match. Both innings are circulary linked with each other with the help of a hyperlink.
  The scorecard of any inning is divided into two parts - Batting and Bowling.
  **Batting** part shows the list of the players who batted in that inning with following attributes - Out type, Runs scored, Balls played, Fours Hit, Sixes Hit, Strike rate.
  **Bowling** part shows the list of the players who bowled in that inning with the attributes - Overs, Balls, Maiden Overs, Runs given, Wickets taken, Economy rate, Dot balls, Fours given, Sixes given, Wides, No balls
- **Stats:**
  This page gives the top batsman and bowler by various options. For batsman one can sort by runs,highest score, strike rate, average,most fours,most sixes,most fifties and most centuries,fastest 50 and fastest 100.
  Bowlers can be sorted by most wickets, maiden, dot balls, average and economy rate.
- **Player Stats:**
  User can search a player by his name and get his batting and bowling statistics.

- **Login:**
  This page allows only privileged (having username and a password) users to log into their accounts. The user only requires a correct set of username and password to log in.
  Regular users can only view data existing on the database while the privilaged users can log into their accounts and add data. The functionality of adding data is explained in the following points.
- **Home:**
  This page is nothing more than a welcome page displaying the text "Welcome ⟨username⟩"
- **Add Team:**
  This page provides the user an option to add a new team to the database. For adding a new team, the form requires only the name of the team. The id of the team is automatically generated with the help of PHP.
- **Add Player:**
  This page provides the user an option to add a new player to the database. For adding a player, the user must enter the same details of a player as the table in the database contains except the id. The id again is automatically generated with the help of PHP.
- **Upload Match:**
  This page provides the user an option to upload the details of an entire match. To upload, the user is provided with an option to upload a file with csv format. The file should contain data in a very specific manner otherwise an error message will show up. The manner in which the data should be present is also available to the user to download in a csv file for his/her reference.
  Note that the details are uploaded in a single transaction with a begin statement at the start and the commit statement at the end. All the details of the match will be ignored if there is an error in any of the insert statements. Either all the details will be inserted in the tables or none of the details will be inserted.
- **Sign Up:**
  An option of sign up is provided inside each login session so that only authorized users can make account for other users. Only username and password is required to sign up. The password is hashed using php function in the script and then stored in the database.

## 2. Special Functionality

- Indexes:
  The indexes are used to scan the tables in the database faster than usual while matching the conditions in the where clause. To facitilate easier use of the database system, postgreSQL automatically builds b-tree index on the primary key of a table while defining the table. Hence, index is already built for those columns which are defined as primary keys in our tables.
  Other than these, we have defined indexes on match_id, striker and bowler in ball table and player_id in player_match table.

```
CREATE INDEX ball_matchid ON ball(match_id);
CREATE INDEX ball_striker ON ball(striker);
CREATE INDEX ball_bowler ON ball(bowler);
CREATE INDEX player_match_playerid ON player_match(player_id);
```

- Views:
  We have created two views namely match_venue which joins match and venue and player_name_match which joins player_match and player.

```
CREATE VIEW match_venue AS SELECT match_id,team1,team2,match_date,season_year,
    venue_name,city_name,country_name,toss_winner,match_winner,toss_name,win_type,
    outcome_type,manofmatch,win_margin FROM match NATURAL JOIN venue;
CREATE VIEW player_name_match AS SELECT player_id,player_name,match_id,player_team
     FROM player_match NATURAL JOIN player;
```

  In addition to the above defined views, we have used views heavily in the back-end of the website to display data. Views created specifically for showing the data are deleted (or dropped) at the end of the php script. Hence, they will not overpopulate the database.

- Triggers:
  We have defined three triggers to implement certain constraints which can not be defined otherwise.
  First trigger is to make sure that the man of the match is playing in the match. Hence, before updating the manofmatch attribute in match, the trigger checks if the player is in player_match table with corresponding match_id.

```
CREATE FUNCTION func_mom() RETURNS trigger as $error_msg$
BEGIN
IF NEW.manofmatch NOT IN (SELECT player_name FROM player_name_match WHERE NEW.
    match_id = player_name_match.match_id) THEN
       RAISE NOTICE 'Man of the match should be playing in the match' USING HINT =
            'Please check player_match table';
       RETURN NULL;
ELSE
       RETURN NEW;
END IF;
END;
$error_msg$ language plpgsql;

CREATE TRIGGER manofmatch_check BEFORE UPDATE OF manofmatch ON match FOR EACH ROW
    EXECUTE PROCEDURE func_mom();
```

  Second trigger is to make sure that striker, non striker, bowler, player out and fielder are playing in the match. So, before inserting or updating the above attributes in ball table, it is checked if they are present in player_match table with corresponding match_id.

```
CREATE FUNCTION func_players() RETURNS trigger as $error_msg$
BEGIN
IF NEW.striker NOT IN (SELECT player_id FROM player_match WHERE NEW.match_id =
    player_match.match_id) AND NEW.striker IS NOT NULL THEN
       RAISE NOTICE 'Striker should be playing in the match' USING HINT = 'Please
           check player_match table';
       RETURN NULL;
ELSIF NEW.non_striker NOT IN (SELECT player_id FROM player_match WHERE NEW.
    match_id = player_match.match_id) AND NEW.non_striker IS NOT NULL THEN
       RAISE NOTICE 'Non stiker should be playing in the match' USING HINT = '
           Please check player_match table';
       RETURN NULL;
ELSIF NEW.bowler NOT IN (SELECT player_id FROM player_match WHERE NEW.match_id =
    player_match.match_id) AND NEW.bowler IS NOT NULL THEN
       RAISE NOTICE 'Bowler should be playing in the match' USING HINT = 'Please
           check player_match table';
       RETURN NULL;
ELSIF NEW.player_out NOT IN (SELECT player_id FROM player_match WHERE NEW.match_id
     = player_match.match_id) AND NEW.player_out IS NOT NULL THEN
       RAISE NOTICE 'Player out should be playing in the match' USING HINT = '
           Please check player_match table';
       RETURN NULL;
ELSIF NEW.fielder NOT IN (SELECT player_id FROM player_match WHERE NEW.match_id =
    player_match.match_id) AND NEW.fielder IS NOT NULL THEN
       RAISE NOTICE 'Fielder should be playing in the match' USING HINT = 'Please
           check player_match table';
       RETURN NULL;
ELSE
       RETURN NEW;
END IF;
RETURN NULL;
END;
$error_msg$ language plpgsql;
```

```
CREATE TRIGGER players_check BEFORE INSERT OR UPDATE OF striker, non_striker,
    bowler, player_out, fielder ON ball FOR EACH ROW EXECUTE PROCEDURE
    func_players();
```

Third trigger makes sure that the admin user (with insert rights) can not include the name of a player twice in player_match table.

```
CREATE FUNCTION func_oneplayer() RETURNS trigger as $error_msg$
BEGIN
IF NEW.player_id IN (SELECT player_id FROM player_match WHERE match_id = NEW.
    match_id) THEN
       RAISE EXCEPTION 'No player can not play in a match twice' USING HINT = '
           Please check player_match table';
       RETURN NULL;
ELSE
       RETURN NEW;
END IF;
RETURN NULL;
END;
$error_msg$ language plpgsql;

CREATE TRIGGER oneplayer_check BEFORE INSERT OR UPDATE OF player_id ON
    player_match FOR EACH ROW EXECUTE PROCEDURE func_oneplayer();
```

- Access Privilages:
  We have created three user modes: Normal, Clerk and Superuser:
  - Normal: This mode is only allowed to query our database using SELECT queries.
  - Clerk: This mode can only insert data to our database. They can't modify or delete previous data.
  - Superuser: The superuser has unlimited access privileges to the database.

## 3. List of Queries

(a) **Teams:**

```
SELECT * FROM team;
```

(b) **Players:**

```
SELECT player_id,player_name,dob,batting_hand,bowling_skill,country_name FROM
    player WHERE country_name IN $country_list;
SELECT player_id,player_name,dob,batting_hand,bowling_skill,country_name FROM
    player WHERE lower(bowling_skill) LIKE '%right%' OR lower(bowling_skill) LIKE
    '%leg%';
SELECT player_id,player_name,dob,batting_hand,bowling_skill,country_name FROM
    player WHERE lower(bowling_skill) LIKE '%left%';
```

(c) **Matches:**

```
CREATE VIEW ptable_allteams_mcount as select distinct team1, count(*) from (
    select team1 from match where season_year = $year union all select team2 from
    match where season_year = $year ) as allmatch group by team1;
CREATE VIEW ptable_nmatches as select min(count) as nmatches, count(distinct team1
    ) as nteams, min(count) * count(distinct team1) /2 as allmatch from
    ptable_allteams_mcount;
CREATE VIEW matches as select * from (select row_number() OVER(order by match_date
    ) AS rownum , * from ptable_nmatches, match where season_year = $year order by
     rownum) as tmatch where rownum <= allmatch;
CREATE VIEW ptable_wins as select match_winner as team, count(*) as wins, count(*)
     * 2 as points from matches where match_winner is not null group by
    match_winner order by points desc;
CREATE VIEW ptable_draws as select distinct team , count(*) as epoints from
    ptable_wins, matches where matches.outcome_type <> 'Result' and matches.
    outcome_type <> 'Superover' and (ptable_wins.team = matches.team1 or
    ptable_wins.team = matches.team2) group by team, matches.nmatches;
```

```
CREATE VIEW ptable1 as select ptable.team, nmatches, wins, nmatches - wins -
    epoints as loss , points + epoints as points from ( select ptable_wins.team,
    nmatches, wins, CASE WHEN epoints IS NULL THEN 0 ELSE epoints END AS epoints,
    points from ptable_nmatches, ptable_wins left join ptable_draws on ptable_wins
    .team = ptable_draws.team) as ptable order by points desc;
CREATE VIEW ptable_nrr_inni as select team_match.match_id as matches_id,
    innings_no as inni , team_name as team_batting from team_match, team, matches
    where team_batting = team.team_id and team_match.match_id = matches.match_id;
CREATE VIEW ptable_nrr_balls as select * from ptable_nrr_inni, ball where
    ptable_nrr_inni.matches_id = ball.match_id and ptable_nrr_inni.inni = ball.
    innings_no;
CREATE VIEW ptable_nrr1 as select team_batting, round((round(sum(runs_scored),2)+
    round(sum(extra_runs),2))/ round(count(ball_id),2)*6 ,2) as nrr1, round(sum(
    runs_scored),2) as runs, round(count(ball_id)/6,2) as overs from
    ptable_nrr_balls group by team_batting;
CREATE VIEW ptable_nrr_inni1 as select team_match.match_id as matches_id,
    innings_no as inni , team_name as team_bowling from team_match, team, matches
    where team_bowling = team.team_id and team_match.match_id = matches.match_id;
CREATE VIEW ptable_nrr_balls1 as select * from ptable_nrr_inni1, ball where
    ptable_nrr_inni1.matches_id = ball.match_id and ptable_nrr_inni1.inni = ball.
    innings_no;
CREATE VIEW ptable_nrr2 as select team_bowling, round((round(sum(runs_scored),2)+
    round(sum(extra_runs),2))/ round(count(ball_id),2)*6 ,2) as nrr2, round(sum(
    runs_scored),2) as runs, round(count(ball_id)/6,2) as overs from
    ptable_nrr_balls1 group by team_bowling;
CREATE VIEW ptable_nrr as select ptable_nrr1.team_batting, nrr1-nrr2 as nrr from
    ptable_nrr1, ptable_nrr2 where ptable_nrr1.team_batting = ptable_nrr2.
    team_bowling order by nrr desc;
SELECT team, nmatches, wins, loss, nrr, points from ptable1, ptable_nrr where team
     = team_batting order by points desc, nrr desc;
DROP VIEW IF EXISTS ptable_nrr, ptable_nrr2, ptable_nrr_balls1, ptable_nrr_inni1,
    ptable_nrr1, ptable_nrr_balls, ptable_nrr_inni, ptable1, ptable_draws,
    ptable_wins, matches, ptable_nmatches, ptable_allteams_mcount;

SELECT * FROM match_venue ORDER BY match_date;
SELECT * FROM match_venue WHERE EXTRACT(year FROM match_date) = '$year' ORDER BY
    match_date;
```

(d) **Match Scorecard(batting):**

```
SELECT team_name from team where team_id = (SELECT team_batting from team_match
    where match_id = $match_id and innings_no =$inn);
CREATE view scard_inni_1_allbat as select distinct player.player_name as Bats_name
     from ball, player where match_id = $match_id and (ball.striker = player.
    player_id or ball.non_striker = player.player_id) and innings_no = $inn;
CREATE view scard_inni_1_bat_p1 as select player.player_name as Bats_name, sum(
    runs_scored) as runs, count(runs_scored) as ball,(round(round(sum(runs_scored)
    ,2)/round(count(runs_scored),2)*100,2)) as srate,striker_batting_position from
     ball, player where match_id = $match_id and ball.striker = player.player_id
    and ( extra_type = 'No Extras' or extra_type = 'legbyes' or extra_type = 'byes
    ' or extra_type = 'noballs' ) and innings_no = $inn group by player.
    player_name, striker_batting_position order by striker_batting_position;
CREATE view scard_inni_1_bat_p2 as select scard_inni_1_allbat.Bats_name ,n4, n6
    from scard_inni_1_allbat left join ( select distinct player_name as pname,
    count(*) as n4 from ball, player where match_id = $match_id and player_id =
    striker and innings_no = $inn and runs_scored = 4 group by pname) as temp2 on
    temp2.pname = scard_inni_1_allbat.Bats_name left join ( select distinct
    player_name as pname, count(*) as n6 from ball, player where match_id =
    $match_id and player_id = striker and innings_no = $inn and runs_scored = 6
    group by pname) as temp3 on temp3.pname = scard_inni_1_allbat.Bats_name;
CREATE view scard_inni_1_bat_p3 as select player_name, out_type from ball, player
    where match_id = $match_id and innings_no = $inn and ((ball.striker = player.
    player_id and out_type <> 'Not Applicable' and out_type <> 'run out') or (ball
    .non_striker = player.player_id and out_type = 'run out'));
```

```
SELECT scard_inni_1_allbat.Bats_name, CASE WHEN out_type IS NULL THEN 'Not Out'
    ELSE out_type END AS out_type,CASE WHEN runs IS NULL THEN 0 ELSE runs END AS
    runs,CASE WHEN ball IS NULL THEN 0 ELSE ball END AS ball, CASE WHEN n4 IS NULL
     THEN 0 ELSE n4 END AS n4,CASE WHEN n6 IS NULL THEN 0 ELSE n6 END AS n6, CASE
    WHEN srate IS NULL THEN 0 ELSE srate END AS srate from scard_inni_1_allbat
    left join scard_inni_1_bat_p3 on scard_inni_1_bat_p3.player_name =
    scard_inni_1_allbat.Bats_name left join scard_inni_1_bat_p2 on
    scard_inni_1_bat_p2.Bats_name = scard_inni_1_allbat.Bats_name left join
    scard_inni_1_bat_p1 on scard_inni_1_bat_p1.Bats_name = scard_inni_1_allbat.
    Bats_name order by scard_inni_1_bat_p1.striker_batting_position;
DROP VIEW IF EXISTS scard_inni_1_bat_p1, scard_inni_1_allbat, scard_inni_1_bat_p2,
     scard_inni_1_bat_p3 CASCADE;
```

(e) **Match Scorecard(bowling):**

```
SELECT team_name from team where team_id = (SELECT team_bowling from team_match
    where match_id = $match_id and innings_no = $inn);
CREATE view scard_bowl_inni1_bowlorder as select distinct bowler, min(over_id) as
    bowl_order from ball where match_id = $match_id and innings_no = $inn group by
     bowler order by bowl_order;
CREATE view scard_bowl_nball_inni1_p1 as select bowler, mod(count(*), 6) as balls
    from ball where match_id = $match_id and innings_no = $inn and ( extra_type =
    'No Extras' or extra_type = 'legbyes' or extra_type = 'byes') group by bowler,
     over_id order by over_id;
CREATE view scard_bowl_nball_inni1_nover as select bowler, count(*) as nover from
    scard_bowl_nball_inni1_p1 where balls = 0 group by bowler;
CREATE view scard_bowl_nball_inni1_nball as select distinct bowler , max(balls) as
     nballs from scard_bowl_nball_inni1_p1 group by bowler;
CREATE view scard_bowl_inni1_runs_p1 as select bowler, sum(extra_runs) as nerun
    from ball where match_id = $match_id and innings_no = $inn and (extra_type = '
    wides' or extra_type = 'noballs') group by bowler;
CREATE view scard_bowl_inni1_runs_p2 as select bowler, sum(runs_scored) as nrun
    from ball where match_id = $match_id and innings_no = $inn group by bowler;
CREATE view scard_bowl_inni1_nwickets as select distinct bowler, count(out_type)
    as nwickets from ball where match_id = $match_id and innings_no = $inn and (
    out_type = 'caught' or out_type = 'Keeper Catch' or out_type = 'caught and
    bowled'or out_type = 'stumped' or out_type = 'lbw' or out_type = 'bowled')
    group by bowler;
CREATE view scard_bowl_inni1_n4 as select distinct bowler as pname, count(*) as n4
     from ball where match_id = $match_id and innings_no = $inn and runs_scored =
    4 group by pname;
CREATE view scard_bowl_inni1_n6 as select distinct bowler as pname, count(*) as n6
     from ball where match_id = $match_id and innings_no = $inn and runs_scored =
    6 group by pname;
CREATE view scard_bowl_inni1_n0 as select distinct bowler as pname, count(*) as n0
     from ball where match_id = $match_id and innings_no = $inn and runs_scored =
    0 and ( extra_type = 'No Extras' or extra_type = 'legbyes' or extra_type = '
    byes') group by pname;
CREATE view scard_bowl_inni1_nwide as select distinct bowler as pname, sum(
    extra_runs) as nwide from ball where match_id = $match_id and innings_no =
    $inn and extra_type = 'wides' group by pname;
CREATE view scard_bowl_inni1_nnoball as select distinct bowler as pname, sum(
    extra_runs) as nnoball from ball where match_id = $match_id and innings_no =
    $inn and extra_type = 'noballs' group by pname;
CREATE view scard_bowl_inni1_nmaiden as select bowler, count(*) as nmaiden from (
    select distinct bowler, count(*), over_id from ball where match_id = $match_id
     and innings_no = $inn and runs_scored = 0 and ( ( extra_runs = 0 and (
    extra_type = 'wides' or extra_type = 'noballs' or extra_type = 'No Extras') )
    or ( extra_type = 'legbyes' or extra_type = 'byes' or extra_type = 'penalty')
    ) group by bowler, over_id) as nmaiden1 where count = 6 group by bowler;
CREATE view scard_bowl_inni1_nevery as select scard_bowl_inni1_bowlorder.bowler,
    CASE WHEN nover IS NULL THEN 0 ELSE nover END AS nover,CASE WHEN nballs IS
    NULL THEN 0 ELSE nballs END AS nballs,CASE WHEN nrun IS NULL THEN 0 ELSE nrun
    END AS nrun,CASE WHEN nerun IS NULL THEN 0 ELSE nerun END AS nerun,CASE WHEN
```

```
    nmaiden IS NULL THEN 0 ELSE nmaiden END AS nmaiden,CASE WHEN nwickets IS NULL
    THEN 0 ELSE nwickets END AS nwickets, CASE WHEN n0 IS NULL THEN 0 ELSE n0 END
    AS n0, CASE WHEN n4 IS NULL THEN 0 ELSE n4 END AS n4,CASE WHEN n6 IS NULL THEN
     0 ELSE n6 END AS n6, CASE WHEN nwide IS NULL THEN 0 ELSE nwide END AS nwide,
    CASE WHEN nnoball IS NULL THEN 0 ELSE nnoball END AS nnoball from
    scard_bowl_inni1_bowlorder left join scard_bowl_inni1_nwickets on
    scard_bowl_inni1_nwickets.bowler = scard_bowl_inni1_bowlorder.bowler left join
     scard_bowl_inni1_n4 on scard_bowl_inni1_n4.pname = scard_bowl_inni1_bowlorder
    .bowler left join scard_bowl_inni1_n6 on scard_bowl_inni1_n6.pname =
    scard_bowl_inni1_bowlorder.bowler left join scard_bowl_inni1_n0 on
    scard_bowl_inni1_n0.pname = scard_bowl_inni1_bowlorder.bowler left join
    scard_bowl_inni1_nwide on scard_bowl_inni1_nwide.pname =
    scard_bowl_inni1_bowlorder.bowler left join scard_bowl_inni1_nnoball on
    scard_bowl_inni1_nnoball.pname = scard_bowl_inni1_bowlorder.bowler left join
    scard_bowl_inni1_nmaiden on scard_bowl_inni1_nmaiden.bowler =
    scard_bowl_inni1_bowlorder.bowler left join scard_bowl_inni1_runs_p1 on
    scard_bowl_inni1_runs_p1.bowler = scard_bowl_inni1_bowlorder.bowler left join
    scard_bowl_inni1_runs_p2 on scard_bowl_inni1_runs_p2.bowler =
    scard_bowl_inni1_bowlorder.bowler left join scard_bowl_nball_inni1_nover on
    scard_bowl_nball_inni1_nover.bowler = scard_bowl_inni1_bowlorder.bowler left
    join scard_bowl_nball_inni1_nball on scard_bowl_nball_inni1_nball.bowler =
    scard_bowl_inni1_bowlorder.bowler;
CREATE view scard_bowl_inni1_nevery1 as select scard_bowl_inni1_nevery.bowler,
    nover, nballs, nrun + nerun as runs_given, nmaiden, nwickets, n0, n4, n6,
    nwide, nnoball from scard_bowl_inni1_nevery;
SELECT player.player_name as player_name, nover , nballs, nmaiden,runs_given,
    nwickets, round(round(runs_given,2)/round(nover*6+nballs,2)*6,2) as econ, n0,
    n4,n6,nwide,nnoball from scard_bowl_inni1_bowlorder, scard_bowl_inni1_nevery1,
     player where player.player_id = scard_bowl_inni1_bowlorder.bowler and
    scard_bowl_inni1_bowlorder.bowler = scard_bowl_inni1_nevery1.bowler order by
    bowl_order;
DROP VIEW IF EXISTS scard_bowl_inni1_bowlorder,scard_bowl_nball_inni1_p1,
    scard_bowl_nball_inni1_nover,scard_bowl_nball_inni1_nball,
    scard_bowl_inni1_runs_p1,scard_bowl_inni1_runs_p2,scard_bowl_inni1_nwickets,
    scard_bowl_inni1_n4,scard_bowl_inni1_n6,scard_bowl_inni1_n0,
    scard_bowl_inni1_nwide,scard_bowl_inni1_nnoball,scard_bowl_inni1_nmaiden,
    scard_bowl_inni1_nevery,scard_bowl_inni1_nevery1 CASCADE;
```

(f) **Stats(batting):**

```
SELECT player_name AS Player,total_matches as Matches,total_runs as Total_Runs,
    highest_score as Highest_Score, Round(average_runs,2) as AVG_Runs ,
    total_balls_faced as Balls_Faced, Round(total_runs*100.0/total_balls_faced,2)
    as Strike_Rate, num_100 as Centuries,num_50 as Fifties,total_num4 as Fours,
    total_num6 as Sixes FROM ( select player_name,striker,sum(runs_per_match) as
    total_runs,sum(balls_faced_permatch) as total_balls_faced, sum(num4_permatch)
    as total_num4, sum(num6_permatch) as total_num6, max(runs_per_match) as
    highest_score, AVG(runs_per_match) as average_runs,count(runs_per_match)
    filter(where runs_per_match>=50 and runs_per_match<100) as num_50, count(
    runs_per_match) filter(where runs_per_match >=100) as num_100 from (select
    player_name, ball.striker, sum(runs_scored) as runs_per_match, count(ball_id)
    as balls_faced_permatch, count(ball_id) filter(where runs_scored=4) as
    num4_permatch, count(ball_id) filter(where runs_scored=6) as num6_permatch
    from ball join player on (ball.striker=player.player_id) group by match_id,
    ball.striker, player.player_name) as x group by striker,player_name ) as
    table1 join (select player_id,count(distinct match_id) as total_matches from
    player_match group by player_id) as table2 on striker=player_id order by
    total_runs DESC limit 100;
```

(g) **Stats(fastest fifty):**

```
CREATE VIEW ball_cumm_runs as select match_id, striker, over_id,ball_id,count(
    ball_id) over w as num_balls,count(ball_id) filter(where runs_scored=6) over w
     as num_6s, count(ball_id) filter(where runs_scored=4) over w as num_4s, sum(
```

```
    runs_scored) over w as cumm_runs from ball window w as (partition by match_id,
    striker order by over_id,ball_id);

SELECT match_id,player.player_name,min(num_balls) as Balls_Faced,min(num_6s) as
    num_6s,min(num_4s) as num_4s,min(cumm_runs) as runs from ball_cumm_runs join
    player on ball_cumm_runs.striker = player.player_id where cumm_runs>=50 group
    by match_id, striker, player.player_name order by Balls_Faced limit 50;
```

(h) **Stats(bowling):**

```
CREATE VIEW stats_bowl_nmatches as select player_name, count(distinct match_id) as
     nmatches from ball, player where bowler = player_id group by player_name;
CREATE VIEW stats_bowl_mwickets as select player_name, count(*) as nwicket from
    ball, player where bowler = player_id and ( out_type <> 'Not Applicable' and
    out_type <> 'retired hurt' and out_type <> 'run out' and out_type <> '
    obstructing the field' ) group by player_name order by nwicket desc;
CREATE VIEW stats_bowl_nmaidens as select player_name, count(*) as nmaiden from (
    select distinct bowler, over_id, count(*) from ball where runs_scored = 0 and
    ( ( extra_runs = 0 and (extra_type = 'wides' or extra_type = 'noballs' or
    extra_type = 'No Extras') ) or ( extra_type = 'legbyes' or extra_type = 'byes'
     or extra_type = 'penalty') ) group by match_id, bowler, over_id) as nmaiden1,
     player where nmaiden1.count = 6 and bowler = player_id group by player_name
    order by nmaiden desc;
CREATE VIEW stats_bowl_ndots as select distinct player_name as pname, count(*) as
    n0 from ball, player where player_id = bowler and runs_scored = 0 and ( (
    extra_runs = 0 and (extra_type = 'wides' or extra_type = 'noballs' or
    extra_type = 'No Extras') ) or ( extra_type = 'legbyes' or extra_type = 'byes'
     or extra_type = 'penalty') ) group by pname order by n0 desc;
CREATE VIEW stats_bowl_runs_p1 as select bowler, sum(extra_runs) as nerun from
    ball where (extra_type = 'wides' or extra_type = 'noballs') group by bowler;
CREATE VIEW stats_bowl_runs_p2 as select bowler, sum(runs_scored) as nrun from
    ball group by bowler;
CREATE VIEW stats_bowl_runs as select player.player_name, (nrun + nerun ) as runs
    from stats_bowl_runs_p1, stats_bowl_runs_p2, player where stats_bowl_runs_p1.
    bowler = stats_bowl_runs_p2.bowler and stats_bowl_runs_p1.bowler = player_id
    order by runs desc;
CREATE VIEW stats_bowl_avg as select stats_bowl_mwickets.player_name, round(round(
    runs,2)/ round(nwicket,2) , 2) as avg from stats_bowl_runs,
    stats_bowl_mwickets where stats_bowl_runs.player_name = stats_bowl_mwickets.
    player_name and nwicket > 10 order by avg;
CREATE VIEW stats_bowl_econ as select stats_bowl_runs.player_name , round(round(
    runs,2)/round(balls,2),2)*6 as econ, balls from ( select bowler, count(*) as
    balls from ball where ( extra_type = 'No Extras' or extra_type = 'legbyes' or
    extra_type = 'byes' or extra_type = 'penalty') group by bowler) as stats_econ,
     stats_bowl_runs, player where stats_bowl_runs.player_name = player.
    player_name and stats_econ.bowler = player.player_id and balls > 120 order by
    econ;
SELECT stats_bowl_mwickets.player_name,nmatches as no_innings, nwicket, nmaiden,
    n0, runs, avg, econ from stats_bowl_mwickets left join stats_bowl_nmaidens on
    stats_bowl_mwickets.player_name = stats_bowl_nmaidens.player_name left join
    stats_bowl_ndots on stats_bowl_mwickets.player_name = stats_bowl_ndots.pname
    left join stats_bowl_runs on stats_bowl_mwickets.player_name = stats_bowl_runs
    .player_name left join stats_bowl_avg on stats_bowl_mwickets.player_name =
    stats_bowl_avg.player_name left join stats_bowl_econ on stats_bowl_mwickets.
    player_name = stats_bowl_econ.player_name, stats_bowl_nmatches where
    stats_bowl_nmatches.player_name = stats_bowl_mwickets.player_name order by
    nwicket desc nulls last limit 50;

DROP VIEW IF EXISTS stats_bowl_econ, stats_bowl_avg, stats_bowl_runs,
    stats_bowl_runs_p2, stats_bowl_runs_p1, stats_bowl_ndots, stats_bowl_nmaidens,
     stats_bowl_mwickets, stats_bowl_nmatches;
```

(i) **Player Stats:**

```
SELECT Player_Name,Player_Id from player where Player_Name ILIKE '%".$_GET['
    search_field']."%';
SELECT count(distinct match_id) as total_matches from player_match where player_id
    ='$player_id';
SELECT sum(runs_scored) as total_runs,count(ball_id) as balls_played,count(ball_id
    ) filter(where runs_scored=4) as num_4,count(ball_id) filter(where runs_scored
    =6) as num_6 from ball where striker=$player_id;
SELECT max(runs_per_match) as highest_score, AVG(runs_per_match) as average_runs,
    count(runs_per_match) filter(where runs_per_match>=50 and runs_per_match<100)
    as num_50, count(runs_per_match) filter(where runs_per_match>=100) as num_100
    from (select sum(runs_scored) as runs_per_match from ball where ball.striker=
    $player_id group by match_id ) as x;
SELECT count(*) as num_over,count(*) filter(where numWicketsPerOver=4) as num_4w,
    count(*) filter(where numWicketsPerOver=3) as num_3w from (select match_id,
    over_id,count(out_type) filter(where out_type Not in ('Not Applicable','
    retired hurt','obstructing the field','hit wicket','run out')) as
    numWicketsPerOver from ball where bowler=$player_id group by match_id,over_id)
     as x;
SELECT sum(runs_scored) + sum(extra_runs) filter(where extra_type NOT In ('penalty
    ')) as total_runs,count(out_type) filter(where out_type Not in ('Not
    Applicable','retired hurt','obstructing the field','hit wicket','run out')) as
     num_wickets,count(ball_id) as num_balls,count(distinct match_id) as
    num_matches from ball where bowler=$player_id;
```

(j) **Login:**

```
SELECT password FROM login WHERE uname = '$usname';
```

(k) **Add Team:**

```
SELECT MAX(team_id) FROM team';
INSERT INTO team VALUES ($id , '$teamname');
```

(l) **Add Player:**

```
SELECT MAX(player_id) FROM player;
INSERT INTO player VALUES ($id, '$pname', '$pdob', '$pbat', '$pbowl', '$pcountry')
    ;
```

(m) **Upload Match:**

```
BEGIN;
SET DATESTYLE to MDY;
INSERT INTO match VALUES($str);
INSERT INTO player_match VALUES($str);
INSERT INTO team_match VALUES($str);
INSERT INTO ball VALUES($str);
COMMIT;
```

(n) **Sign Up:**

```
INSERT INTO login VALUES('$usname','$pwd')
```

4. **Query Times**

| Query Number | Running Time (ms) |
|---|---|
| (a) | 1.109 |
| (b) | 2.518 |
| (c) | 93.769 |
| (d) | 30.494 |
| (e) | 53.028 |
| (f) | 149.617 |
| (g) | 243.473 |
| (h) | 574.978 |
| (i) | 5.924 |
| (j) | 0.585 |
| (k) | 10.819 |
| (l) | 12.342 |
| (m) | 156.183 |
| (n) | 5.319 |

# 4 ER Diagram

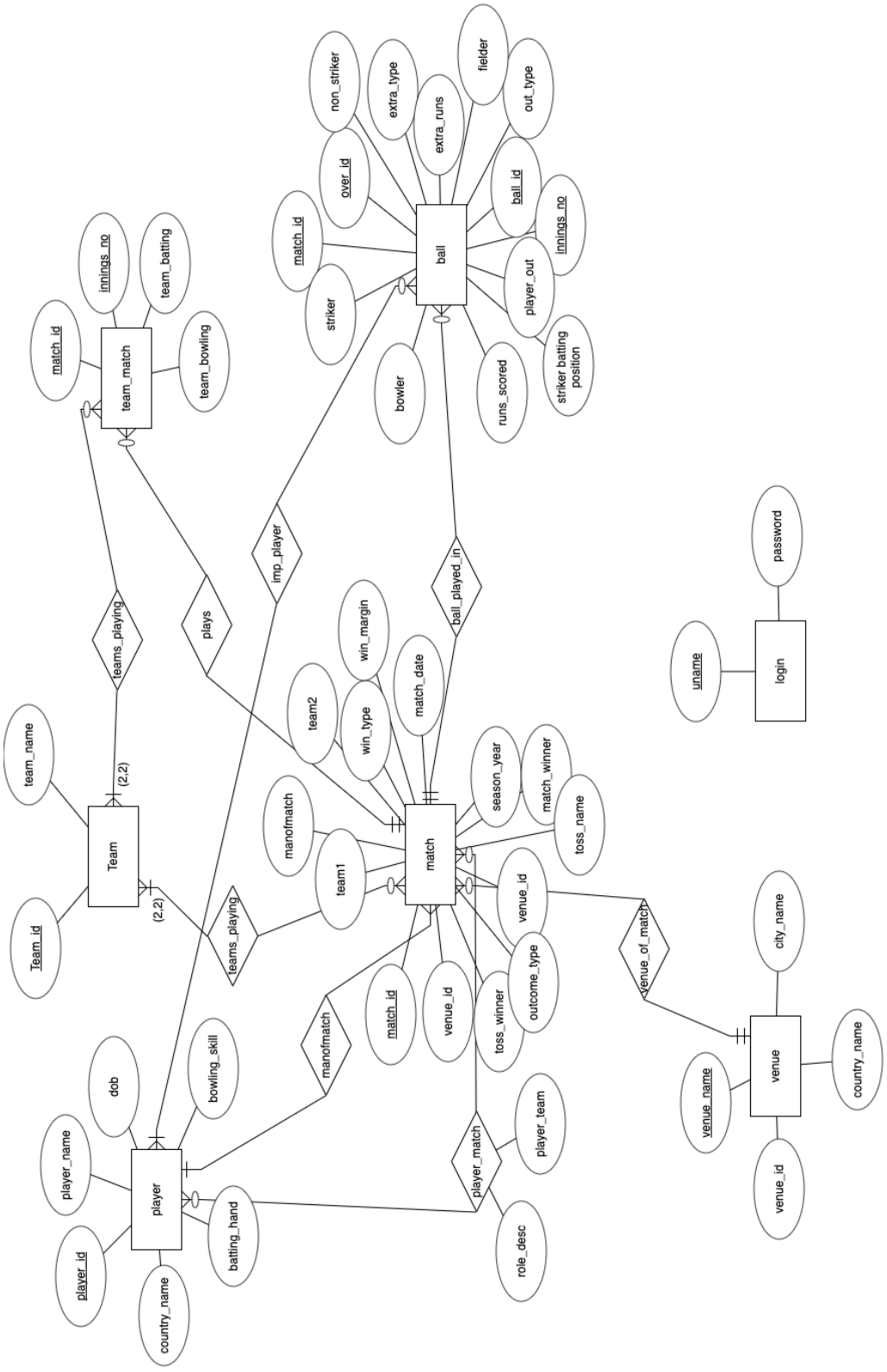


Figure 1: ER Diagram