

**Project# 2**

**Submitted by:**

**Sarthak Raste (010107331)**

**Keertikeya Gupta**

**Amit Fernandes (010106564)**

**Izhar Raazi**

**CmPE- 226**

**Fall 2015**

**Submitted to:**

**Dr. Kong Li**

**Index**

1. **Team Contribution and Timeline**
2. **Introduction**
3. **Technologies Used**
4. **Design**
5. **Implementation**
6. **Test Plan**
7. **Analysis**

**Team Contributions:**

Team #4:

|  |  |
| --- | --- |
| **Name** | **Contribution** |
| Keertikeya Gupta | Project proposal, ER diagram, UI Design and Implementation, Connectivity |
| Sarthak Raste | ER diagram, DB Script, Documentation |
| Amit Fernandes | ER diagram, DB Script, Documentation |
| Izhar Raazi | ER diagram, DB Script, Documentation |

**Timeline:**

**ERD**: 10th Sept– 17th Sept 2015

**DB Script**: 10th Nov – 16th Nov 2015

**UI Design**: 18th Nov – 25th Nov 2015

**Connectivity**: 26th Nov- 28th Nov 2015

**Documentation**: 28th Nov – 29th Nov 2015

**Introduction:**

To make a database management system for sports statistics for the game of Cricket. This system will allow users to view the individual statistics of players, match details, stats of teams and other stats such as leader-boards and records. Administrator of the system can manage the data within the system, and perform functions such as addition, update, and deletion of data.

**Mini-world Description**

The mini-world for this system consists of various Cricket playing teams, players and other people who are involved in the game (such as match officials, team support staff, etc.). Other entities are Matches that will be played by teams and their players, and venues that will host different matches between teams.

**Actors**

1. Players
2. Teams
3. Venue
4. Match
5. Officials
6. Team Support

**Scenarios**

For all scenarios, let us assume the following:

A match is being played between two teams, Team1 and Team2. Each team consists of a squad of 15 players for each match. Of these 15 players, there are 11 active players, the remaining 4 are for substitution.

The match is hosted in Stadium1 (Venue), home ground for Team1. Stadium1 is located in City1 and has a capacity of X.

Scenario 1:

Team1 bats first, scores R runs in O1 overs at a loss of Wkt1 wickets. Team2 scores (R+1) runs in O2 overs at a loss of Wkt2 wickets. Result: Team2 wins by W wickets.

Scenario 2:

Team1 bats first, scores R1 runs in O1 overs at a loss of Wkt1 wickets. Team2 scores R2 runs (R1 > R2) in O2 overs at the loss of Wkt2 wickets. Result: Team1 wins by (R1 – R2) runs.

Scenario 3:

Team1 bats first, scores R runs in O1 overs at a loss of Wkt1 wickets. Team2 also scores R runs in O2 overs at a loss of Wkt2 wickets. Result: Match drawn.

Scenario 4:

Rain interrupts match. Result: Match abandoned due to rain.

**Functionalities and operations of actors:**

1. **Player:**

Batsman – a) Scores runs (by hitting a boundary (4 or 6 runs) and/or by running between the wickets).

b) Gets out (bowled, caught, stumped, runout).

Bowler – a) Bowls multiple overs (6 valid balls each).

b) Takes wickets of batsmen.

Fielder – a) Restricts the batsmen from taking runs.

b) Takes wickets of batsmen by catching or running them out.

Substitutes - Take place of tired or injured players during fielding .

1. **Team:**

Play matches against other teams. Win/loose a match.

1. **Match:**

Played between two teams. Hosted at a venue.

1. **Venue:**

Host matches between two teams at a time.

1. **Officials:**

Umpires: Give decisions about wickets and runs

Commentators: Provide live analysis of ongoing matches

1. **Team support:**

Sponsors - provide funding for team

Coach - Coaching and guiding team players

Team manager - Manage team, players, equipment and match schedules

Team selectors - Select team squad and the playing 11, appoint captain and vice-captain

**Technologies Used:**

**Database Engine- InnoDB (MySQL)**

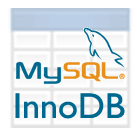
**Languages/Tools : Java, Swing, JDBC, NetBeans, WAMP Server.**

**InnoDB:** InnoDB is a storage engine for MySQL. MySQL 5.5 and later use it by default.[1] It provides the standard ACID-compliant transaction features, along with foreign key support (Declarative Referential Integrity). It is included as standard in most binaries distributed by MySQL AB, the exception being some OEM versions.

**JDBC:** Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, that defines how a client may access a database. It is part of the Java Standard Edition platform, from Oracle Corporation.

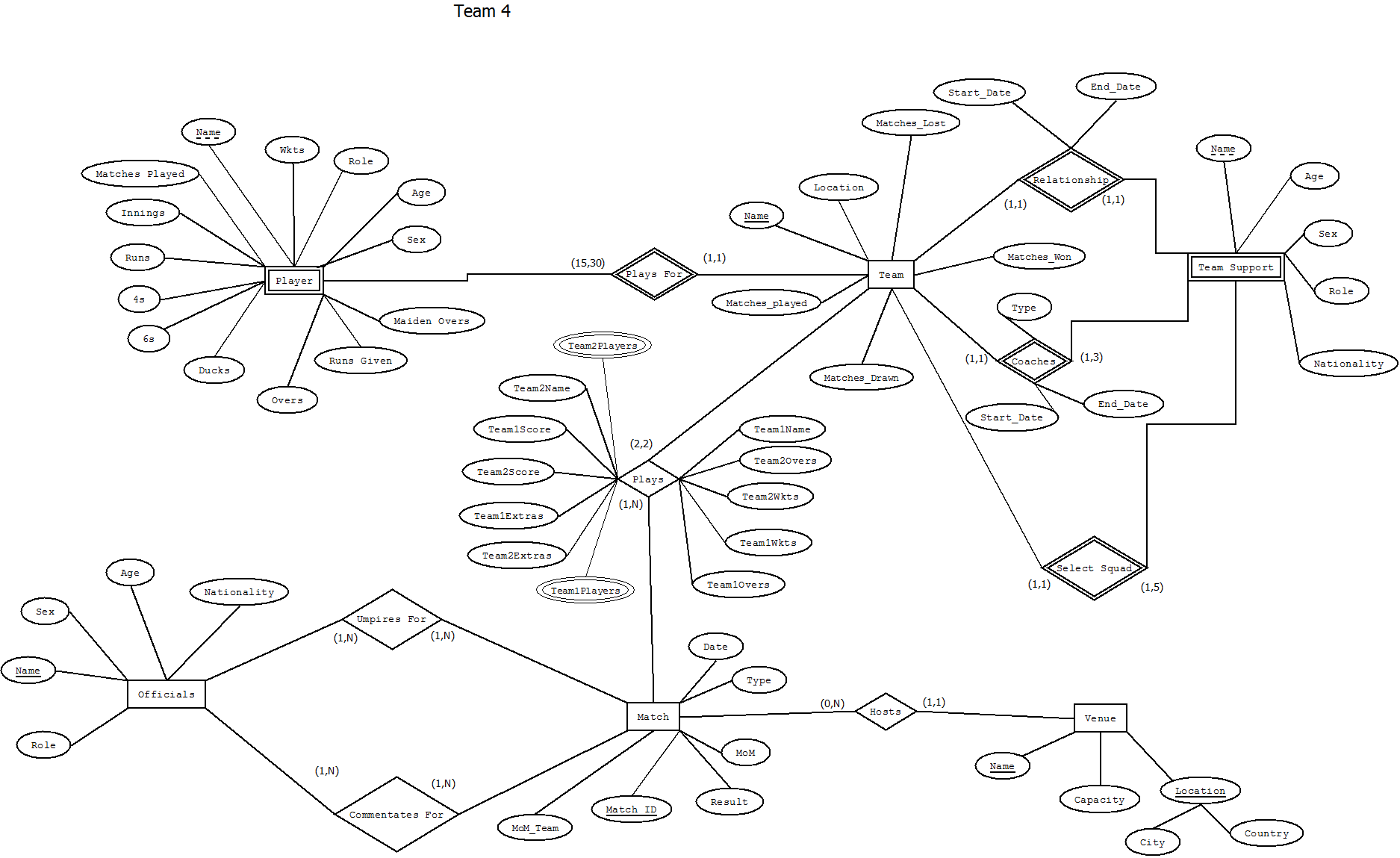
**WAMP Server:** A Windows Web development environment for Apache, MySQL, PHP databases.

**NetBeans:** NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform, including the NetBeans integrated development environment (IDE), can be extended by third party developers.

****

****

****

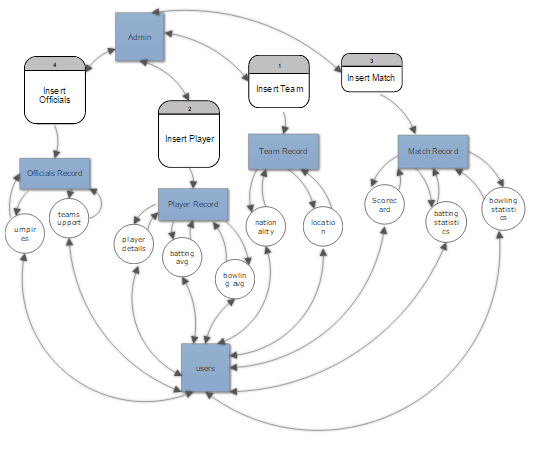
**Design:**

**Changes done in ERD:**

No first name and last name in player  
Role is not multivalued  
Four new attributes added to team - matches played, won, lost and drawn  
In match, add man of the match and team.

**Reason**: Since these changes will help in improving the system

**Flow of data (DFD):**



**Normal form of 3NF has been achieved for all tables. No table/relationship is below 3NF**

**Components of the system:**

**Data:** It is a very important component of the database system. Most of the organizations generate, store and process 1arge amount of data. The data acts a bridge between the machine parts i.e. hardware and software and the users which directly access it or access it through some application programs.

**Hardware:** The hardware consists of the secondary storage devices such as magnetic disks (hard disk, zip disk, floppy disks), optical disks (CD-ROM), magnetic tapes etc. on which data is stored together with the Input/Output devices (mouse, keyboard, printers), processors, main memory etc. which are used for storing and retrieving the data in a fast and efficient manner. Since database can range from those of a single user with a desktop computer to those on mainframe computers with thousand of users, therefore proper care should be taken for choosing appropriate hardware devices for a required database.

**Software:** The Software part consists of DBMS which acts as a bridge between the user and the database or in other words, software that interacts with the users, application programs, and database and files system of a particular storage media (hard disk, magnetic tapes etc.) to insert, update, delete and retrieve data. For performing these operations such as insertion, deletion and updation we use SQL.

**Users:** Users are those persons who need the information from the database to carry out their primary business responsibilities i.e. Personnel, Staff, Clerical, Managers, Executives etc. On the basis of the job and requirements made by them they are provided access to the database totally or partially.

**Data, Hardware, Software and Users interact with each other to achieve the database system.**

**Trade-offs:**

In order to achieve consistency and isolation, the system does not support concurrent transactions.

Idempotent operations are not supported to maintain atomicity of transactions.

**Implementation: SQL Queries**

CREATE TABLE IF NOT EXISTS `Users`(

`Username` VARCHAR(20),

`Password` VARCHAR(20),

`Role` VARCHAR(10)

);

**Description : DML Query to create table for users.**

CREATE TABLE IF NOT EXISTS `Player` (

`Name` varchar(50) NOT NULL ,

`Wkts` int(11) DEFAULT 0,

`Role` varchar(30) NOT NULL DEFAULT 0,

`Dob` date DEFAULT '1950-01-01',

`Balls` int(5) default 0,

`BatAvg` decimal(6,3) DEFAULT 0,

`BatStrRate` decimal(6,3) DEFAULT 0,

`Maidens` int(11) DEFAULT 0,

`RunsGiven` int(11) DEFAULT 0,

`Overs` int(11) DEFAULT 0,

`Ducks` int(11) DEFAULT 0,

`6s` int(11) DEFAULT 0,

`4s` int(11) DEFAULT 0,

`BatHigh` int(3) DEFAULT 0,

`BowlAvg` decimal(6,3) DEFAULT 0.0,

`bowlecon` decimal(6,3) DEFAULT 0.0,

`MoM` int(3) DEFAULT 0,

`Runs` int(11) DEFAULT 0,

`Innings` int(11) DEFAULT 0,

`MatchesPlayed` int(11) DEFAULT 0,

`PlaysFor` varchar(11) NOT NULL,

PRIMARY KEY (`Name`,`PlaysFor`)

);

**Description : DML Query to create table for players.**

CREATE TABLE IF NOT EXISTS `Team` (

`Name` varchar(25) NOT NULL ,

`Location` varchar(25) NOT NULL,

`MatchesPlayed` int(5) default 0,

`MatchesWon` int(5) default 0,

`MatchesLost` int(5) default 0,

`MatchesDrawn` int(5) default 0,

`winp` decimal(5,2) default 0.0,

PRIMARY KEY (`Name`)

);

**Description : DML Query to create table for team.**

CREATE TABLE IF NOT EXISTS `Plays` (

`MatchID` int(11),

`T1Name` varchar(25) NOT NULL ,

`T2Name` varchar(25) NOT NULL,

`T2Overs` decimal(3,1) NOT NULL DEFAULT 0.0,

`T1Overs` decimal(3,1) NOT NULL DEFAULT 0.0,

`T1Score` int(2) NOT NULL DEFAULT 0,

`T2Score` int(2) NOT NULL ,

`T1Wkts` int(2) NOT NULL DEFAULT 0,

`T2Wkts` int(2) NOT NULL DEFAULT 0,

`T1Extras` int(2) NOT NULL DEFAULT 0,

`T2Extras` int(2) NOT NULL DEFAULT 0,

PRIMARY KEY (`MatchID`, `T1Name`, `T2Name`)

);

**Description : DML Query to create table for plays.**

CREATE TABLE IF NOT EXISTS `Match` (

`MatchId` int(11) NOT NULL AUTO\_INCREMENT,

`Date` Date NOT NULL,

`Type` varchar(8) NOT NULL,

`MoM` varchar(50) NOT NULL,

`MoMTeam` varchar(25) NOT NULL,

`Result` varchar(50) NOT NULL,

`Venue` varchar(40) NOT NULL,

PRIMARY KEY (`MatchId`)

);

**Description : DML Query to create table for match.**

CREATE TABLE IF NOT EXISTS `Officials` (

`Name` varchar(40) NOT NULL,

`Sex` varchar(10) NOT NULL,

`DoB` date NOT NULL DEFAULT '1950-01-01',

`Nationality` varchar(25) NOT NULL,

`Role` varchar(20) NOT NULL,

PRIMARY KEY (`Name`)

);

**Description : DML Query to create table for officials.**

CREATE TABLE IF NOT EXISTS `Venue` (

`Name` varchar(40) NOT NULL,

`City` varchar(20) NOT NULL,

`Country` varchar(25) NOT NULL,

`Capacity` int(10) NOT NULL DEFAULT 0,

PRIMARY KEY (`Name`,`City`,`Country`)

);

**Description : DML Query to create table for venue.**

CREATE TABLE IF NOT EXISTS `TeamSupport` (

`Name` varchar(40) NOT NULL,

`Sex` varchar(2) NOT NULL,

`DoB` date NOT NULL DEFAULT '1950-01-01',

`Nationality` varchar(25) NOT NULL,

`Role` varchar(20) NOT NULL,

PRIMARY KEY (`Name`)

);

**Description : DML Query to create table for team support.**

CREATE TABLE IF NOT EXISTS `UmpiresFor` (

`UmpireName` varchar(50),

`MatchID` int(11),

PRIMARY KEY(`UmpireName`, `MatchID`)

);

**Description : DML Query to create table for umpires for.**

CREATE TABLE IF NOT EXISTS `CommentatesFor` (

`CommentatorName` varchar(50),

`MatchID` int(11),

PRIMARY KEY(`CommentatorName`, `MatchID`)

);

**Description : DML Query to create table for commentates for.**

CREATE TABLE IF NOT EXISTS `Coaches` (

`CoachName` varchar(50),

`TeamName` varchar(25),

`StartDate` date NOT NULL,

`EndDate` date,

PRIMARY KEY(`CoachName`, `TeamName`)

);

**Description : DML Query to create table for coaches.**

CREATE TABLE IF NOT EXISTS `TeamSelectorFor` (

`SelectorName` varchar(50),

`TeamName` varchar(25),

`StartDate` date NOT NULL,

`EndDate` date,

PRIMARY KEY(`SelectorName`, `TeamName`)

);

**Description : DML Query to create table for team selectors for.**

ALTER TABLE `Player`

ADD FOREIGN KEY (PlaysFor) REFERENCES `Team`(Name)

on delete cascade

on update cascade;

**Description : DML Query to alter player table.**

ALTER TABLE `Plays`

ADD FOREIGN KEY (MatchID) REFERENCES `Match`(MatchID)

on delete cascade

on update cascade,

ADD FOREIGN KEY (T1Name, T2Name) REFERENCES `Team`(Name, Name)

on delete cascade

on update cascade

;

**Description : DML Query to alter plays table.**

ALTER TABLE `Match`

ADD FOREIGN KEY (Venue) REFERENCES `Venue`(Name)

on delete cascade

on update cascade,

ADD FOREIGN KEY (T1Name, T2Name) REFERENCES `Team`(Name, Name)

on delete cascade

on update cascade;

**Description : DML Query to alter table for match.**

ALTER TABLE UmpiresFor

ADD FOREIGN KEY (UmpireName) REFERENCES `Officials`(Name)

on delete cascade

on update cascade,

ADD FOREIGN KEY (MatchID) REFERENCES `Match`(MatchID)

on delete cascade

on update cascade;

**Description : DML Query to alter table for umpires for.**

ALTER TABLE CommentatesFor

ADD FOREIGN KEY (CommentatorName) REFERENCES `Officials`(Name)

on delete cascade

on update cascade,

ADD FOREIGN KEY (MatchID) REFERENCES `Match`(MatchID)

on delete cascade

on update cascade;

**DML Query to alter table for commentates for**

ADD FOREIGN KEY (CoachName) REFERENCES `TeamSupport`(Name)

on delete cascade

on update cascade,

ADD FOREIGN KEY (TeamName) REFERENCES `Team`(Name)

on delete cascade

on update cascade;

**Description : DML Query to add foreign key to teamSupport.**

ALTER TABLE TeamSelectorFor

ADD FOREIGN KEY (SelectorName) REFERENCES `TeamSupport`(Name)

on delete cascade

on update cascade,

ADD FOREIGN KEY (TeamName) REFERENCES `Team`(Name)

on delete cascade

on update cascade;

**Description : DML Query to alter table for TeamSelectorFor.**

create TRIGGER innumsix BEFORE INSERT ON Player

FOR EACH ROW SET @num = @num + NEW.`6s`;

**Description : DML Query to create trigger to calculate most number of 6’s.**

create TRIGGER upnumsix BEFORE UPDATE ON Player

FOR EACH ROW SET @num = @num + NEW.`6s`;

delimiter $$

**Description : DML Query to create trigger.**

create procedure updatePlayerBat()

begin

update player

set batavg = runs/innings, batstrrate = runs/balls;

end $$

delimiter ;

**Description : DML Query to create procedure to calculate batting rate.**

delimiter $$

create procedure updatePlayerBowl()

begin

update player

set bowlavg = runsgiven/wkts, bowlecon = runsgiven/overs;

end $$

delimiter ;

**Description : DML Query to create procedure to calculate bowling average.**

delimiter $$

create procedure toptotalruns()

begin

create view ttr as

select name, playsfor, runs

from player

order by runs desc

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate top total runs.**

delimiter $$

create procedure batavg()

begin

create view ba as

select name, playsfor, batavg

from player

order by batavg desc

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate batting average.**

delimiter $$

create procedure highindi()

begin

create view hi as

select name, playsfor, bathigh

from player

order by bathigh desc

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate highest runs scored.**

delimiter $$

create procedure strrate()

begin

create view sr as

select name, playsfor, batstrrate

from player

order by batstrrate

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate batting strike rate.**

delimiter $$

create procedure mostwickets()

begin

create view mw as

select name, playsfor, wkts

from player

order by wkts

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate most wickets taken.**

delimiter $$

create procedure econrate()

begin

create view er as

select name, playsfor, bowlecon

from player

order by bowlecon

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate bowling economy.**

delimiter $$

create procedure bowlingavg()

begin

create view boa as

select name, playsfor, bowlavg

from player

order by bowlavg

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate bowling average.**

delimiter $$

create procedure maidenovers()

begin

create view mo as

select name, playsfor, maidens

from player

order by maidens

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate maiden overs.**

delimiter $$

create procedure mostwinsteams()

begin

create view mwt as

select name, matcheswon

from team

order by matcheswon

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate most matches won.**

delimiter $$

create procedure mostwinpteams()

begin

create view mwpt as

select name, winp

from team

order by winp

limit 10;

end$$

delimiter ;

**Description : DML Query to create procedure to calculate most wins per team.**

delimiter $$

create procedure winpteam()

begin

update team

set winp=matcheswon/matchesplayed;

end$$

delimiter ;

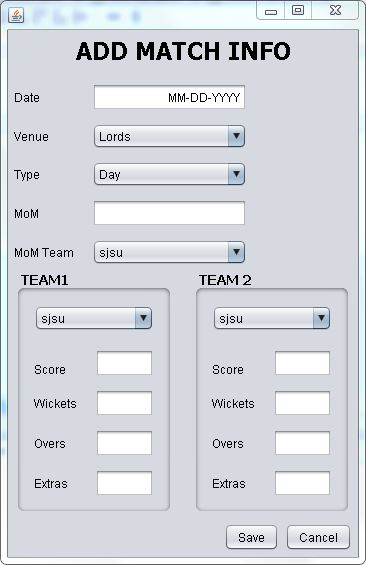
**Description : DML Query to create procedure to calculate win percentage.**

#Added index for query performance

ALTER TABLE Player ADD INDEX (name);

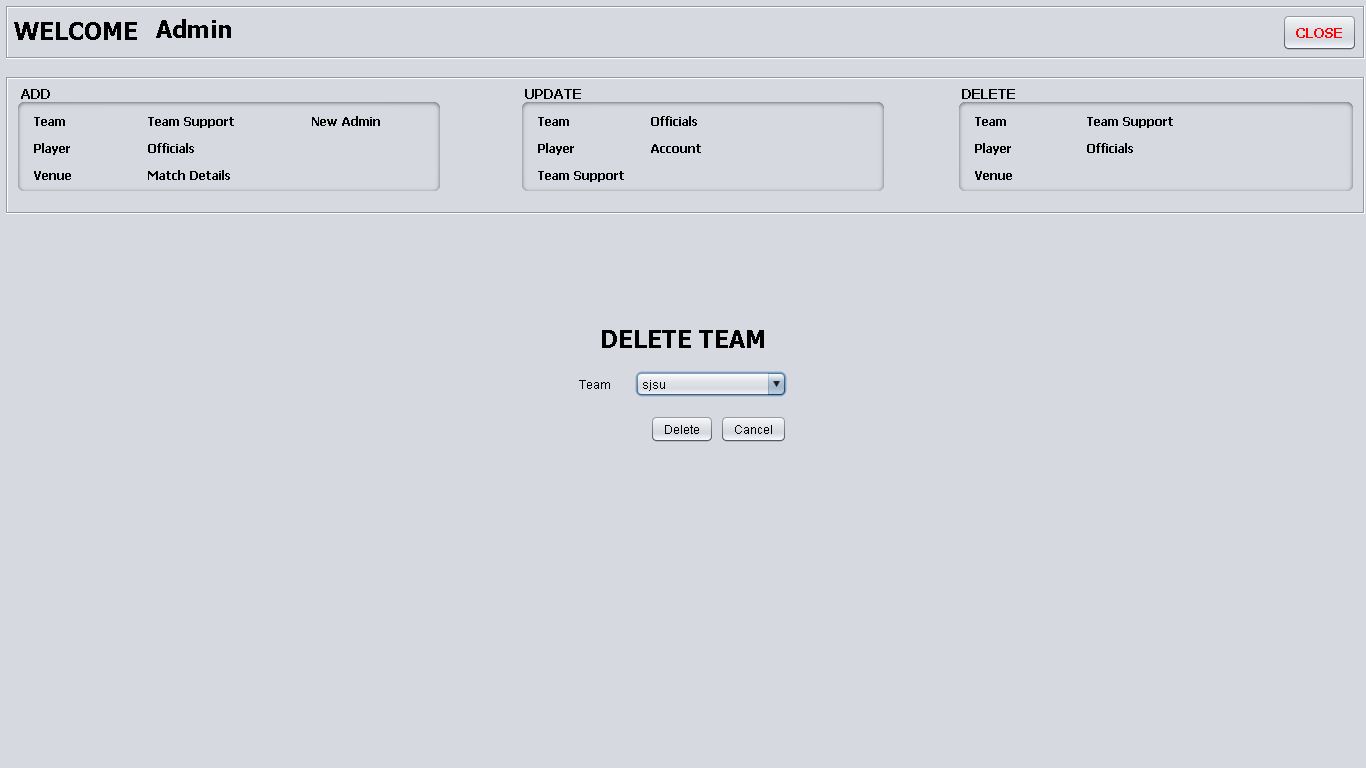
**Description : DML Query to alter table player to improve performance.**

**Screenshot of important operations:**

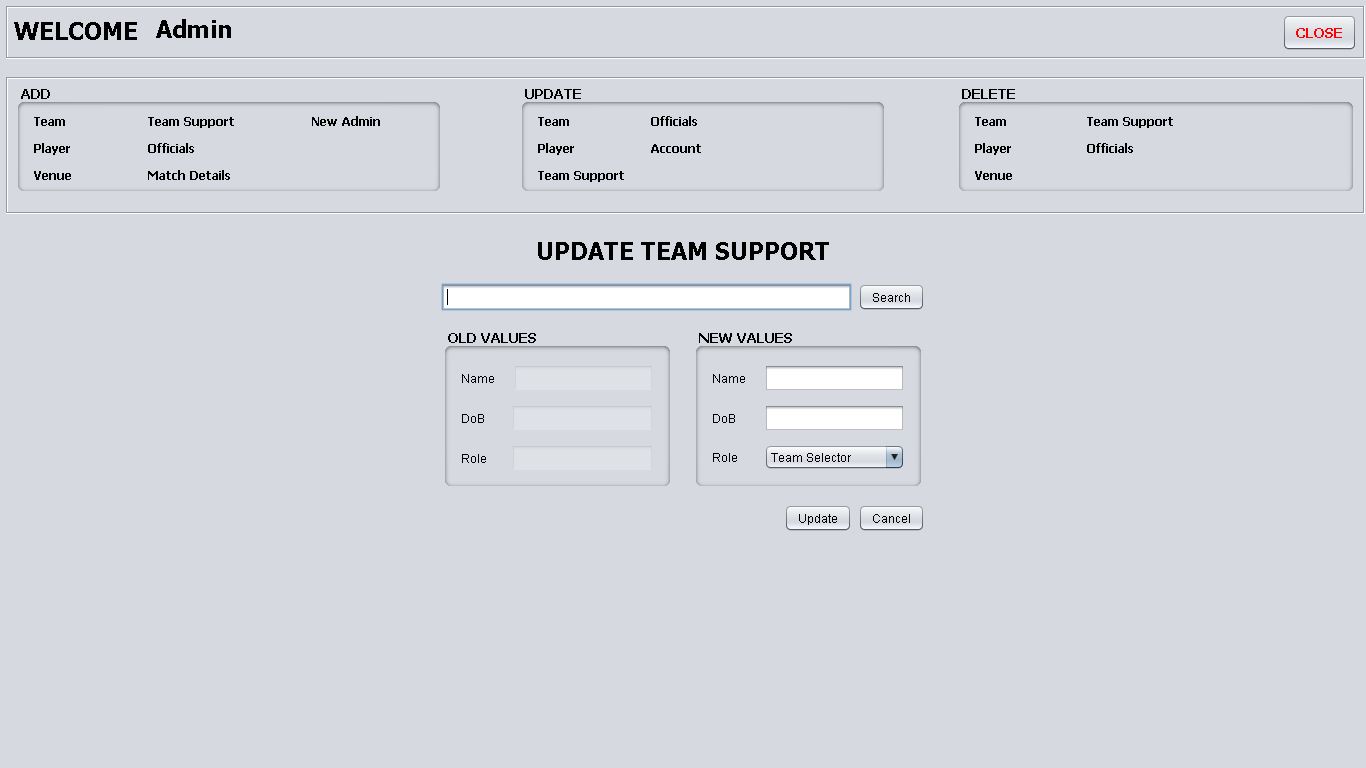
****

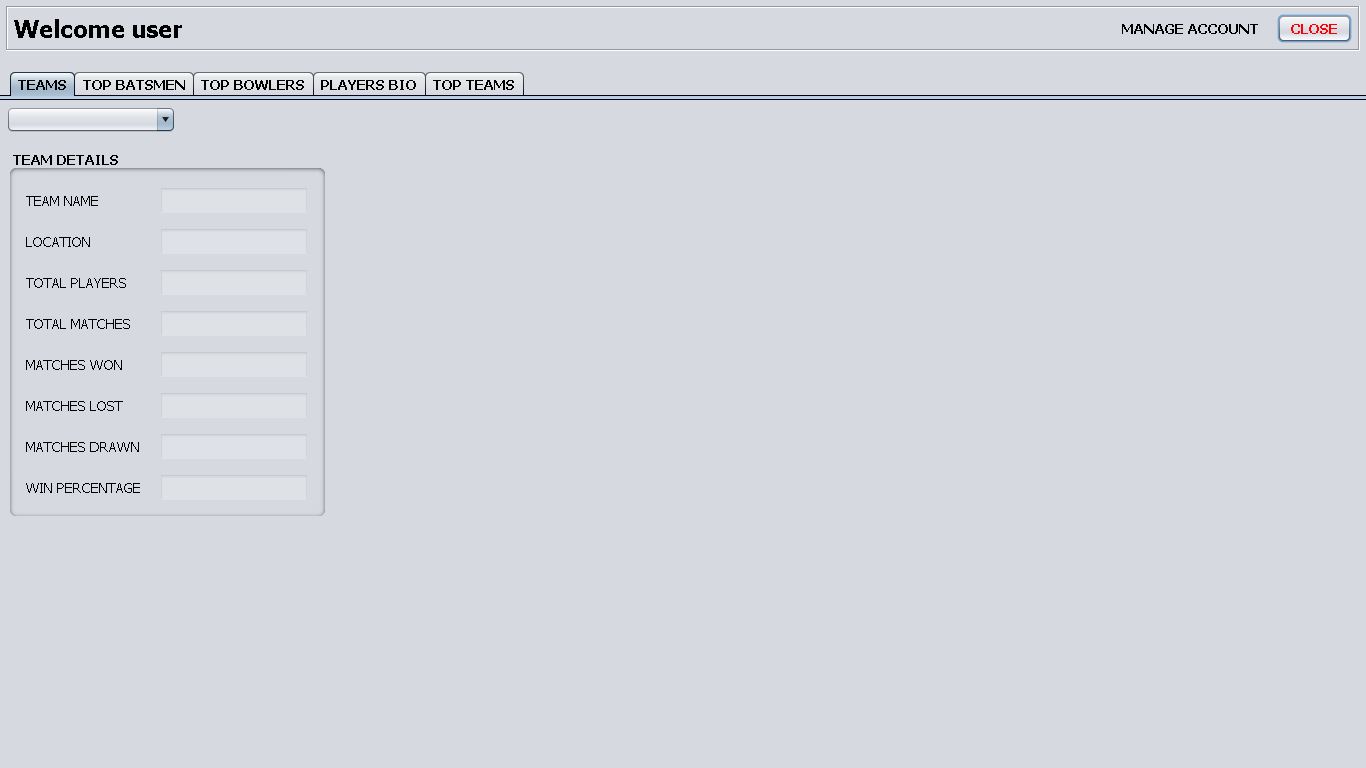
****

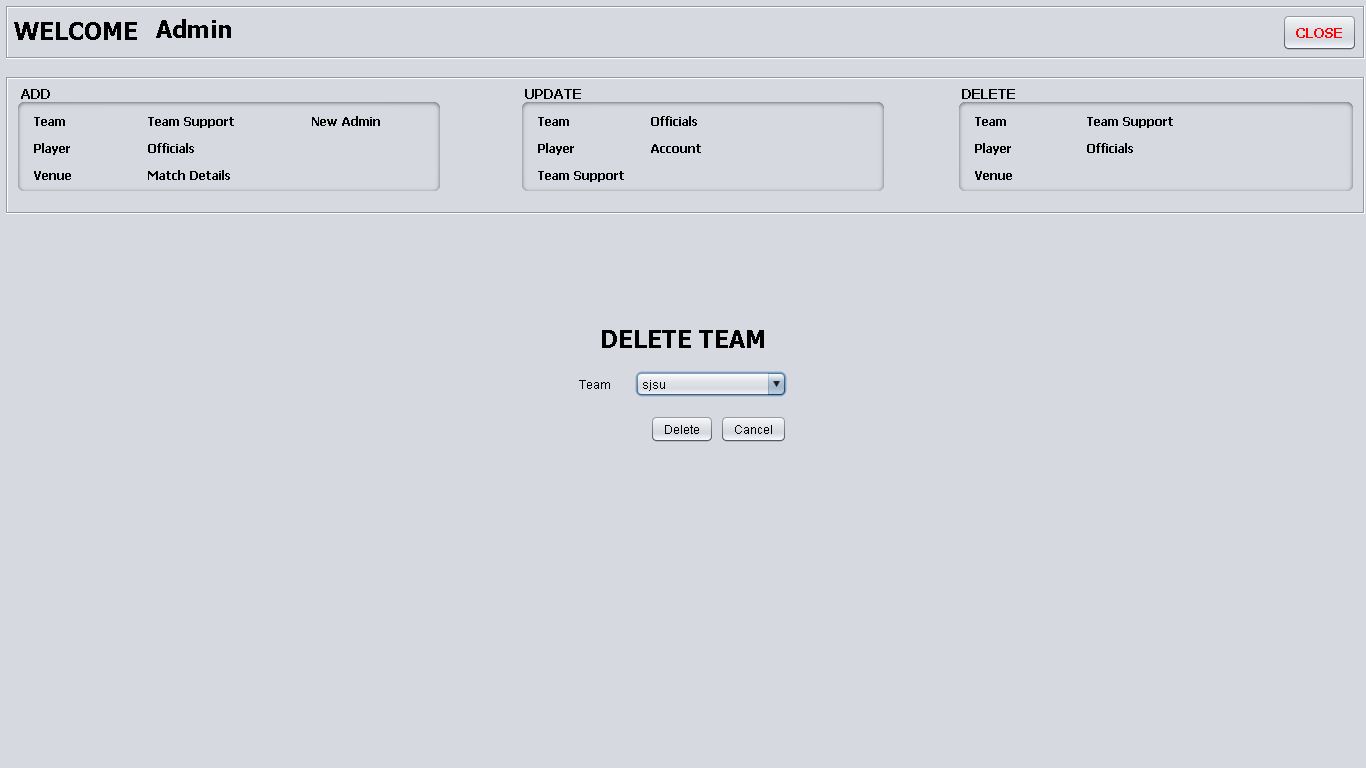
****

****

****

****

****

****

**Testing Plan:**

|  |  |  |
| --- | --- | --- |
| **Test** | **Test Data** | **Expected Results** |
|  | Open the database application | You should be asked to enter a password |
| 2) | Enter the incorrect password | Invalid password dialogue box should appear |
| 3) | Enter the correct password | You will be entered into the database and view the main switchboard either as an admin or a user. |
|  |  |  |
|  | **Logged in as Admin** | **Expected Results** |
| 4) | Create a new data entry | User allowed to create new records |
| 5) | Read data | User allowed to read data |
| 6) | Update a data entry | User allowed to update record only if primary key value is unique |
| 7) | Delete a data entry | User allowed to delete record |
|  |  |  |
|  | **Logged in as User** | **Expected Results** |
| 8) | Read data | User allowed to read data |

**Analysis :**

**Potential Improvements:**

* This system could be further improved by making it distributed.
* A Smartphone application can make use of this RDBMS.
* Data analysis can be achieved in this system by deploying a NoSQL database for large volumes of data.
* A RESTFUL Web based application can be achieved using the database of this system.

**Problems Encountered:**

* Problems faced during Connectivity i.e. integration of a few components of the system.
* Unavailability of a reliable API to get relevant data.
* Mapping of certain object like triggers.