Cricket Ground Infrastructure Management Database

UNIVERSITY OF NORTH TEXAS COLLEGE OF INFORMATION

GROUP 20 TEAM PROJECT REPORT

**Cricket Ground Infrastructure Management Database**

**by**

**Sai Teja Rayabarapu** (Student ID: 11637122)

[*SaiTejaRayabarapu@my.unt.edu*](mailto:SaiTejaRayabarapu@my.unt.edu)

**Major**: Information Science

**Jyothirmai Rapala** (Student ID: 11641526)

*JyothirmaiRapala@my.unt.edu*

**Major**: Computer Engineering

**Instructor’s Name:** Abdullah Alenezi

**Course Title:** Data Modeling for Information Professionals

**Number:** INFO 5707 Section 004 – (Fall 2023)

Date: 12/04/2023

**Database Overview**

**Description/Scope:**

**Vision:**

The vision of the "Cricket Ground Infrastructure Management Database" is to establish a centralized and comprehensive resource that caters to the information management needs of the Board of Control for Cricket in India (BCCI). The database aims to streamline decision-making, enhance historical tracking, and optimize event planning related to cricket grounds.

**Mission:**

The mission is to provide the BCCI with a robust system capable of managing diverse data related to cricket grounds, including infrastructure details, historical records, and facility conditions. The database intends to facilitate efficient ground selection, offer a valuable historical reference, and contribute to data-driven decision-making for successful cricket events.

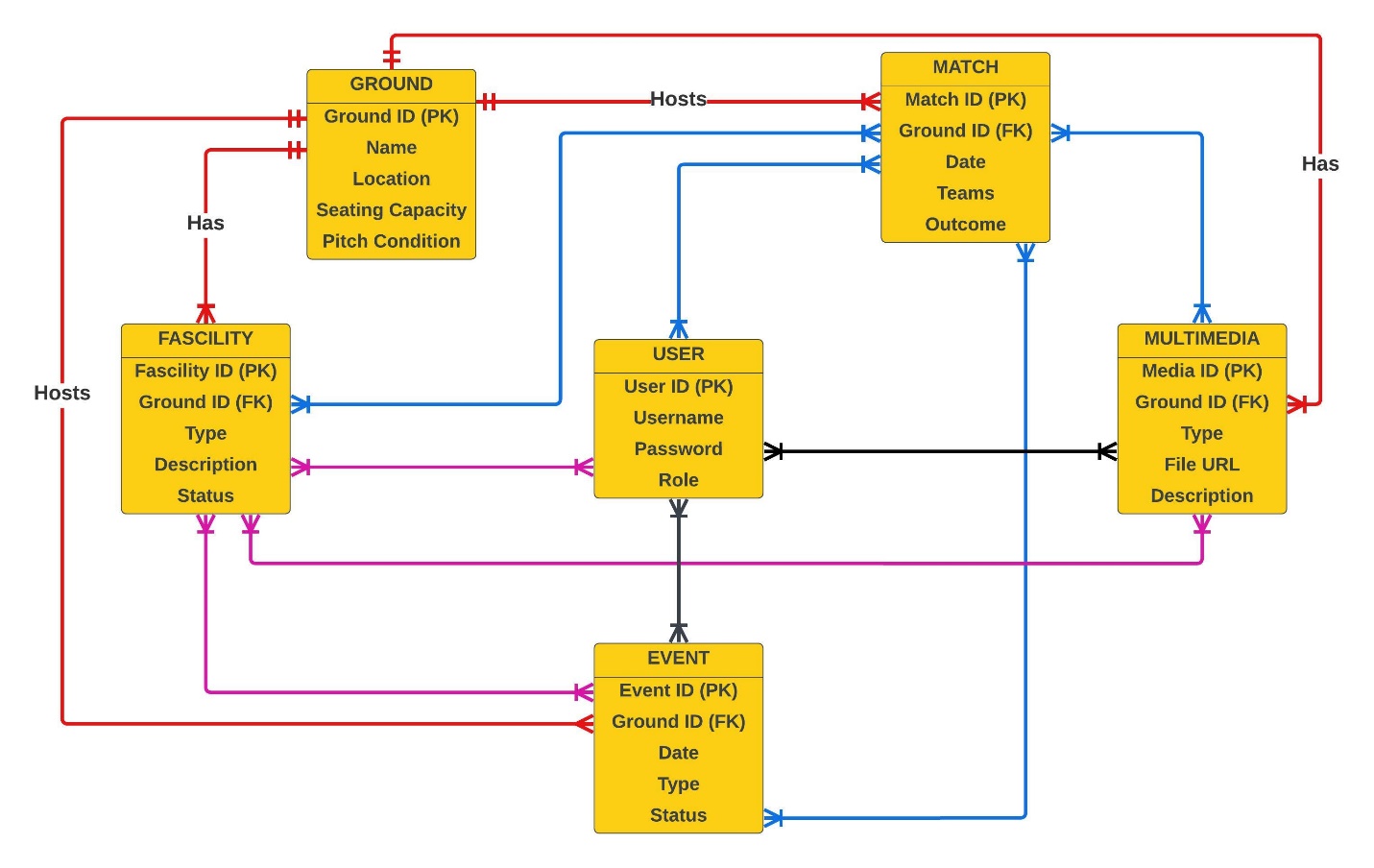
**User Requirements:**

1. **Historical Tracking:** Users should be able to retrieve historical data on cricket grounds, including past matches, records, and significant events.
2. **Infrastructure Management:** Users must have the capability to update and access information about facilities, seating capacity, pitch conditions, and more.
3. **Search and Filter:** The system should provide search and filter options to help users quickly find specific ground information.
4. **Mobile-Friendly:** The database system must be accessible on various devices, including mobile phones, for on-the-go access.
5. **Security:** Access control features should ensure that only authorized users can update and modify the database.
6. **Multimedia Support:** The system must support the attachment of images, videos, or multimedia content related to cricket grounds.
7. **Multilingual Support:** The database should accommodate multiple languages to cater to a global audience.
8. **Collaboration:** Multiple users should be able to use the system simultaneously for collaborative purposes.
9. **Event Planning:** The system should assist in planning future events by providing insights into ground suitability for different formats of the game.
10. **Efficient Operations:** The database should contribute to the overall efficiency of BCCI operations, from match scheduling to event organization.

**Business Rules:**

1. **Access Control:** Only authorized personnel within the BCCI should have access to modify and update database records.
2. **Data Accuracy:** All data entries, especially historical records, must be accurate and verified to ensure the reliability of information.
3. **Regular Updates:** Facilities and infrastructure information should be regularly updated to reflect the current state of cricket grounds.
4. **Language Standardization:** A standard language should be established for data entry to maintain consistency in the database.
5. **Media Rights Management:** Multimedia content uploaded to the system must comply with media rights regulations and permissions.
6. **Event Approval Process:** There should be a defined process for approving cricket events based on the suitability of selected grounds.
7. **Collaborative Editing Rules:** Rules should be in place to manage collaborative editing to avoid conflicts and data inconsistencies.
8. **Backup and Recovery:** Regular backup procedures must be in place to ensure data recovery in case of system failures.
9. **Data Archiving:** Historical records should be archived systematically to maintain database performance.
10. **User Training:** Users should undergo training to understand the database system and its functionalities for optimal usage.

**Entity-Relationship Diagram (ERD) - Crow's Foot Model:**



**Data Dictionary:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Table Name** | **Attribute Name** | **Description** | **Data Type** | **Data Format** | **Required** | **PK or FK** | **Example** |
| Ground | Ground ID | Ground ID | Integer | 999 | Yes | PK | 101 |
| Ground | Name | Name of Ground | CHAR | xxxxxxx | Yes |  | MOTERA |
| Ground | Location | City Name | CHAR | xxxxxxx | Yes |  | AHMEDADABAD |
| Ground | Seating Capacity | Number of Seats | Integer | 9999999 | Yes |  | 103000 |
| Ground | Pitch Condition | How is the pitch? Is it ready for the game | CHAR | xxxxxxx | Yes |  | GOOD AND DRY |
| Match | CricketMtach ID | Mtach ID | Integer | 9 | Yes | PK | 1 |
| Match | Ground ID | Ground ID | Integer | 999 | Yes | FK | 101 |
| Match | Date | Date | CHAR | yy-mm-dd | Yes |  | 10/30/2023 |
| Match | Teams | Team Names | CHAR | xxxxxxx | Yes |  | India and Australia |
| Match | Outcome | Win or Loss | CHAR | xxxxxxx | Yes |  | Win |
| Facility | Facilty ID | Facilty ID | Integer | 999-999 | Yes | PK | 350-151 |
| Facility | Ground ID | Ground ID | Integer | 999 | Yes | FK | 111 |
| Facility | Type | Which Facility | CHAR | xxxxxxx | Yes |  | Practice Pitch |
| Facility | Description | Details about that facility | CHAR | xxxxxxx | Yes |  | Always available |
| Facility | Status | Availble or not | CHAR | xxxxxxx | Yes |  | Yes |
| User | User ID | User ID | Integer | 9999 | Yes | PK | 1111 |
| User | Username | Username | CHAR | xxxxxxx | Yes |  | Vishwesh |
| User | Password | Password | VARCHAR | XXx9999# | Yes |  | [Vishwesh@99](mailto:Vishwesh@99) |
| User | Role | What is the person doing? | CHAR | xxxxxxxx | Yes |  | Curator |
| Multimedia | Media ID | Media ID | Integer | 99 | Yes | PK | 1 |
| Multimedia | Ground ID | Ground ID | Integer | 999 | Yes | FK | 101 |
| Multimedia | Type | Format of the multimedia content | Integer | XXXXXXXX | Yes |  | Video, photo |
| Multimedia | File URL | Reference to the location or path where the multimedia file is stored | CHAR | xxxxxxxx | Yes |  | cricbuzz |
| Multimedia | Description | Textual description or caption associated with the multimedia content | CHAR | xxxxxxxxx | Yes |  | Live Coverage |
| Event | Event ID | Event ID | Integer | 999999 | Yes | PK | 11111 |
| Event | Ground ID | Ground ID | Integer | 999 | Yes | FK | 101 |
| Event | Date | Date | Integer | yy-mm-dd | Yes |  | 10/30/2023 |
| Event | Type | Which type of event? | CHAR | xxxxxxx | Yes |  | Cricket Match |
| Event | Status | Event Confirmed or not | CHAR | xxx | Yes |  | Yes |

**1. Entity Generation and Data Entry:**

* 1. **Database Creation**

**A screen shot of a computer

Description automatically generated**

This SQL code first checks if a database named "CricketDB" exists, and if not, it creates it using the "CREATE DATABASE IF NOT EXISTS" statement. Subsequently, the "USE CricketDB" statement selects "CricketDB" as the active database for subsequent SQL operations within the current session.

* 1. **Table Creation**
     1. **Ground Table**

**A white background with blue and orange text

Description automatically generated**

This SQL code creates a table named "Ground" with five columns: "GroundID" (integer, primary key), "Name" (variable character up to 255 characters), "Location" (variable character up to 255 characters), "SeatingCapacity" (integer), and "PitchCondition" (variable character up to 50 characters). The "CREATE TABLE IF NOT EXISTS" statement ensures that the table is only created if it does not already exist, preventing errors if the table has been created previously. The table is designed to store information about sports grounds, such as their ID, name, location, seating capacity, and pitch condition.

* + 1. **CricketMatch Table**

**A screen shot of a computer code

Description automatically generated**

This SQL code creates a table named "CricketMatch" with six columns: "MatchID" (integer, primary key), "GroundID" (integer), "Date" (date), "Teams" (variable character up to 255 characters), and "Outcome" (variable character up to 50 characters). The table includes a foreign key constraint on the "GroundID" column, establishing a relationship with the "Ground" table's "GroundID" column. This constraint ensures that values in "CricketMatch.GroundID" must correspond to existing values in "Ground.GroundID," enforcing referential integrity between the two tables. The "CREATE TABLE IF NOT EXISTS" statement prevents errors by creating the table only if it does not already exist.

* + 1. **Facility Table**

**A screen shot of a computer code

Description automatically generated**

This SQL code creates a table named "Facility" with five columns: "FacilityID" (integer, primary key), "GroundID" (integer), "Type" (variable character up to 50 characters), "Description" (text), and "Status" (variable character up to 50 characters). The table includes a foreign key constraint on the "GroundID" column, establishing a relationship with the "Ground" table's "GroundID" column. This constraint ensures that values in "Facility.GroundID" must correspond to existing values in "Ground.GroundID," enforcing referential integrity between the two tables. The "CREATE TABLE IF NOT EXISTS" statement ensures that the table is only created if it does not already exist, preventing errors if the table has been created previously.

* + 1. **User Table**

**A close up of text

Description automatically generated**

This SQL code creates a table named "User" with four columns: "UserID" (integer, primary key), "Username" (variable character up to 50 characters), "Password" (variable character up to 50 characters), and "Role" (variable character up to 50 characters). The "UserID" serves as the primary key for uniquely identifying each user record. The table is designed to store user information, including their ID, username, password (hashed or encrypted for security), and role within a system or application. The "CREATE TABLE IF NOT EXISTS" statement ensures that the table is only created if it does not already exist, preventing errors if the table has been created previously.

* + 1. **Multimedia Table**

**A screenshot of a computer code

Description automatically generated**

This SQL code creates a table named "Multimedia" with five columns: "MediaID" (integer, primary key), "GroundID" (integer), "Type" (variable character up to 50 characters), "FileURL" (variable character up to 255 characters), and "Description" (text). The "MediaID" serves as the primary key for uniquely identifying each multimedia record. The table includes a foreign key constraint on the "GroundID" column, establishing a relationship with the "Ground" table's "GroundID" column, ensuring that values in "Multimedia.GroundID" must correspond to existing values in "Ground.GroundID," maintaining referential integrity. The "CREATE TABLE IF NOT EXISTS" statement ensures that the table is only created if it does not already exist, preventing errors if the table has been created previously.

* + 1. **Event Table**

**A screen shot of a computer code

Description automatically generated**

This SQL code creates a table named "Event" with five columns: "EventID" (integer, primary key), "GroundID" (integer), "Date" (date), "Type" (variable character up to 50 characters), and "Status" (variable character up to 50 characters). The "EventID" serves as the primary key for uniquely identifying each event record. The table includes a foreign key constraint on the "GroundID" column, establishing a relationship with the "Ground" table's "GroundID" column, ensuring that values in "Event.GroundID" must correspond to existing values in "Ground.GroundID," maintaining referential integrity. The "CREATE TABLE IF NOT EXISTS" statement ensures that the table is only created if it does not already exist, preventing errors if the table has been created previously.

**1.3 Data Entry**

**1.3.1 Ground Table**

**A screenshot of a computer code

Description automatically generated**

This SQL code uses the "ALTER TABLE" statement to reset the auto-increment counter for the "GroundID" column in the "Ground" table to start from 1. This action ensures that any subsequent insertions into the "Ground" table will assign auto-incremented IDs starting from the specified value, allowing for a controlled sequence of primary key values.

This SQL code uses the "INSERT INTO" statement to add multiple records into the "Ground" table. Each record includes values for the columns "GroundID," "Name," "Location," "SeatingCapacity," and "PitchCondition." The specified values insert information about different sports grounds, such as their unique IDs, names, locations, seating capacities, and pitch conditions, into the "Ground" table in a single database operation.

**1.3.2 CricketMatch Table**

**A white background with black and orange text

Description automatically generated**

This SQL code uses the "INSERT INTO" statement to add records into the "CricketMatch" table. Each record includes values for the columns "MatchID," "GroundID," "Date," "Teams," and "Outcome." The specified values insert information about different cricket matches, such as their unique IDs, associated ground IDs, dates, competing teams, and match outcomes, into the "CricketMatch" table. Additionally, the commented "-- Add more records as needed" suggests that more records can be added following the same format to include additional match information.

**1.3.3 Facility Table**

**A white background with black and orange text

Description automatically generated**

This SQL code utilizes the "INSERT INTO" statement to insert records into the "Facility" table. Each record includes values for the columns "FacilityID," "GroundID," "Type," "Description," and "Status." The specified values provide information about different facilities associated with sports grounds, such as unique IDs, the corresponding ground IDs, facility types, descriptions, and statuses (whether the facility is active or inactive). The commented "-- Add more records as needed" suggests that additional records can be included to extend the information about various facilities.

**1.3.4 User Table**

**A close-up of a computer screen

Description automatically generated**

This SQL code uses the "INSERT INTO" statement to add records into the "User" table. Each record includes values for the columns "UserID," "Username," "Password," and "Role." The specified values insert information about different users, such as unique IDs, usernames, hashed or encrypted passwords, and their roles within a system (e.g., Administrator or RegularUser) into the "User" table. The commented "-- Add more records as needed" suggests that additional user records can be included to expand the user database.

**1.3.5 Multimedia Table**

**A close-up of a computer code

Description automatically generated**

This SQL code utilizes the "INSERT INTO" statement to add records into the "Multimedia" table. Each record includes values for the columns "MediaID," "GroundID," "Type," "FileURL," and "Description." The specified values insert information about different multimedia assets associated with sports grounds, such as unique IDs, corresponding ground IDs, media types (e.g., Image or Video), file URLs, and descriptive information. The commented "-- Add more records as needed" suggests that additional records can be included to incorporate more multimedia assets into the "Multimedia" table.

**1.3.6 Event Table**

**A screen shot of a computer

Description automatically generated**

This SQL code uses the "INSERT INTO" statement to add records into the "Event" table. Each record includes values for the columns "EventID," "GroundID," "Date," "Type," and "Status." The specified values insert information about different events associated with sports grounds, such as unique IDs, corresponding ground IDs, event dates, types (e.g., T20 Tournament or Test Match), and statuses (e.g., Scheduled or Completed) into the "Event" table. The commented "-- Add more records as needed" suggests that additional records can be included to expand the information about various events.

**Overall Result of Entity Generation and Data Entry:**

**A screenshot of a computer

Description automatically generated**

**2. Data Retrieval and Simple Reports:**

**A screenshot of a computer program

Description automatically generated**

The above SELECT statements retrieve all columns and records from the "Ground," "CricketMatch," "Facility," "User," "Multimedia," and "Event" tables, respectively, providing a comprehensive overview of the data in each table.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

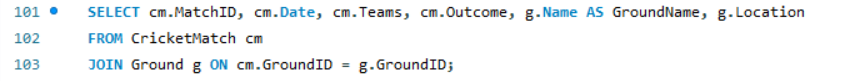
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**2.1 Match Details Report**

****

Provide details of cricket matches, including match ID, date, teams, outcome, ground name, and location.

A screenshot of a computer

Description automatically generated

**2.2 User Role Count Report**

**A close up of words

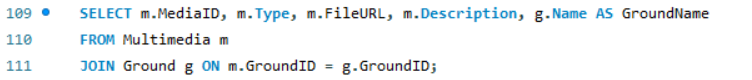
Description automatically generated**

Count the number of users based on their roles.

A screenshot of a computer

Description automatically generated

**2.3 Multimedia Details Report**

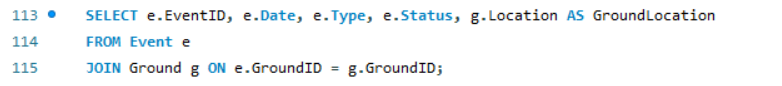
****

Retrieve DE Retrieve all information from the Ground table. Ails of multimedia content, including media ID, type, file URL, description, and ground name.

A screenshot of a computer

Description automatically generated

**2.4 Event Details Report**

****

Retrieve details of events, including event ID, date, type, status, and ground location.

A screenshot of a computer

Description automatically generated

**5 Data Analysis/Report Statements My Database Can Answer**

**1. Match Outcome and Multimedia Association**

**Statement in Plain English:** Find the matches where TeamA won and retrieve multimedia associated with the winning grounds.

**A close-up of a computer screen

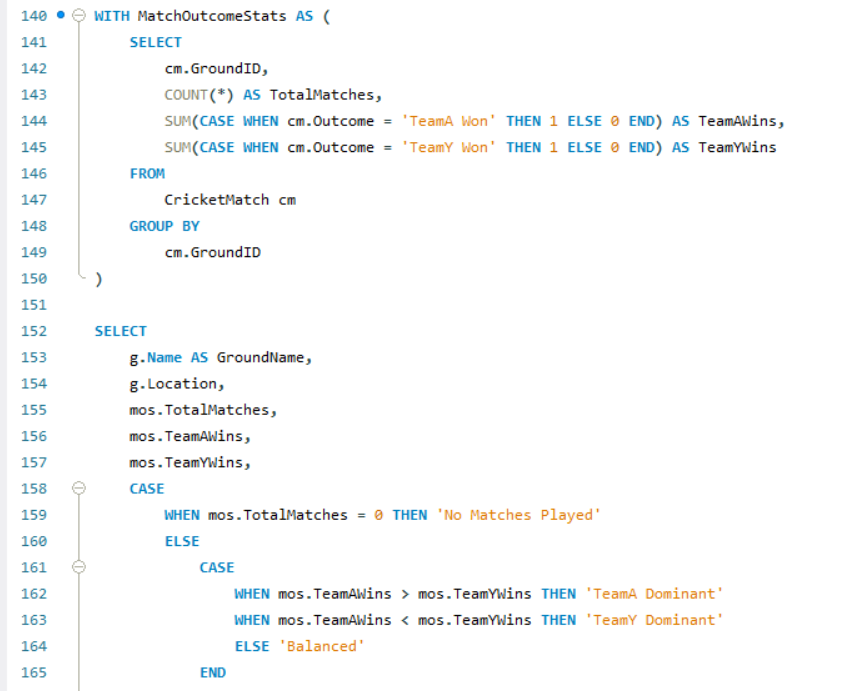
Description automatically generated**

**A screenshot of a computer

Description automatically generated**

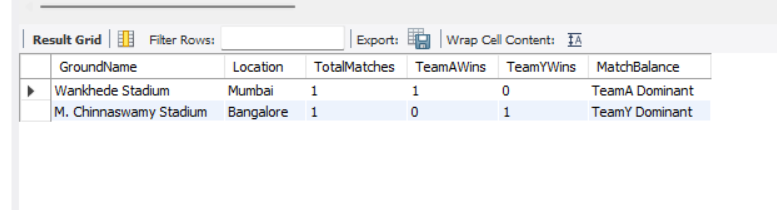
**2. Complex Match Outcome Analysis**

**Statement in Plain English:** This SQL query, named "Complex Match Outcome Analysis," computes detailed statistics on cricket match outcomes for each ground. It includes the total number of matches played, the number of wins for TeamA, the number of wins for TeamY, and an assessment of the overall match balance for each ground. The results are ordered based on the total number of matches played in descending order. This complex analysis provides valuable insights into the performance and competitiveness of matches held at different cricket grounds.



A screen shot of a computer

Description automatically generated

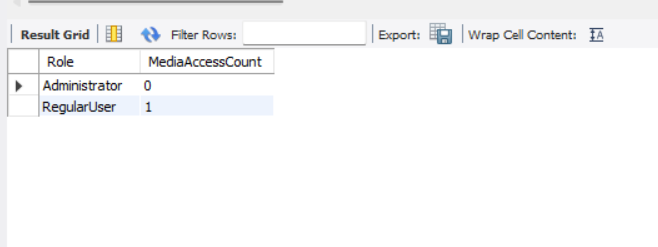


**3. User Role Distribution with Multimedia Access**

**Statement in Plain English:** Determine the distribution of user roles and the count of multimedia items accessed by each role.

A close-up of a computer screen

Description automatically generated

****

**4. Upcoming Events with Available Multimedia**

**Statement in Plain English:** Retrieve details of scheduled events that have associated multimedia content available.

A close-up of a computer screen

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**5. Average Seating Capacity for Each Match Outcome**

**Statement in Plain English:** Calculate the average seating capacity for matches with different outcomes.

A computer code with black text

Description automatically generated

**A screenshot of a computer

Description automatically generated**