“ESports League Management Database”

Team: Group3 (Ferguson, Jaykant, Tyson, Samuel)

Database Theory

CS3520 X02 | 2025 Spring

Final Project

UTAH VALLEY UNIVERSITY

UVU

Table of Contents

Executive Summary3

* What the company or department does

Vision and Objective3

* The objective of the company or department

Mission Statement3

* What is the business solution that the company or company ‘s department offers?

Service3

* The objective of the company or company’s Department

Use Case3

* What is the use case for this solution?

Business requirement4

* Describe the business requirements.

Conceptual Model Diagram5

* Diagram
* Diagram description (Explain the diagram)

Entity-Relation Model Diagram6-10

* Diagram
* Diagram Description (Explain the diagram, Cardinality, PK, FK, and indexes)

The approach used to develop your proposed solution11

Current releases11

* Description of the first release

Future releases12

* Definition of new features, releases, and improvements

Conclusion12

Appendix13

* Populated tables……………………………………………………………………………………………………13-19
  + Data Sample …………………………………………………………………………………………… 20-30
* SQL Scripts Used………………………………………………………………………………………………….30-36
  + Data types
  + Tables created
  + Data population
  + Example
* Time Log of Each Members

**Executive Summary:**

Esport league Organization the main motive of this company is managing online game Events. This organization brings together teams, players, sponsors, and leagues. Operates the Esports events smoothly and creates a new level of experience for all gamers.

Vision and Objective:

Our Vision is to bring Esport Community together by providing a centralized system that handles multiple operations on the Esport League Management.

The objective of the Esport League Management Database is to help organizations optimize operations, attract sponsorships, and grow their competitive presence. It reduces administrative workload to ensure smooth league management, improve decision-making, and promote fair play across all events.

Mission Statement

**Service**

1. Organizing Events like tournaments and leagues.
2. Provide the registration and administration.
3. Provide the real time Match scheduling
4. Track the player compliance and disciplinary actions.

Use Case

Use Case 1: Handle the Esport league and tournament

Use Case 2: Track the Team performance statistics.

Use Case 3. Monitor infraction and game restrictions

Use Case 4. Opportunities for sponsors

Business requirement

1. Player management

* The database can store information about players such as their name, rank, and age
* In order to participate in events, players must belong to a team (even if that team is just a single player)

1. Event and player registration

* Players need to be able to register their teams and pay fees through the online portal
* Players must be able to register themselves through the online portal as well

1. Event record keeping

* The database needs to be able to track information about matches, such as the teams that played in them, the results, and the games being played
* Event sponsors should also be catalogued in case the billing department needs them in future

1. Organize league information

* We should be able to track the state of the league and the placement of teams in the league

1. Issue Citations and Enforce Bans
   * Each player needs a record of infractions so that the organization can take appropriate actions when a player crosses predetermined league thresholds
   * The date of each infraction needs to be recorded
2. Present supported games
   * The organization will be putting together events for a number of different games, and being able to show them to potential players is vital for the health of the leagues
   * Players should be able to find the games by title, genre, or by games appearing in upcoming events
3. Match information and recordings
   * Match replay data should be stored in the database
   * Other match information such as start and end time, winner, and game statistics should be available for each match

Conceptual Model Diagram

A diagram of a flowchart

AI-generated content may be incorrect.

**Conceptual Model Diagram**

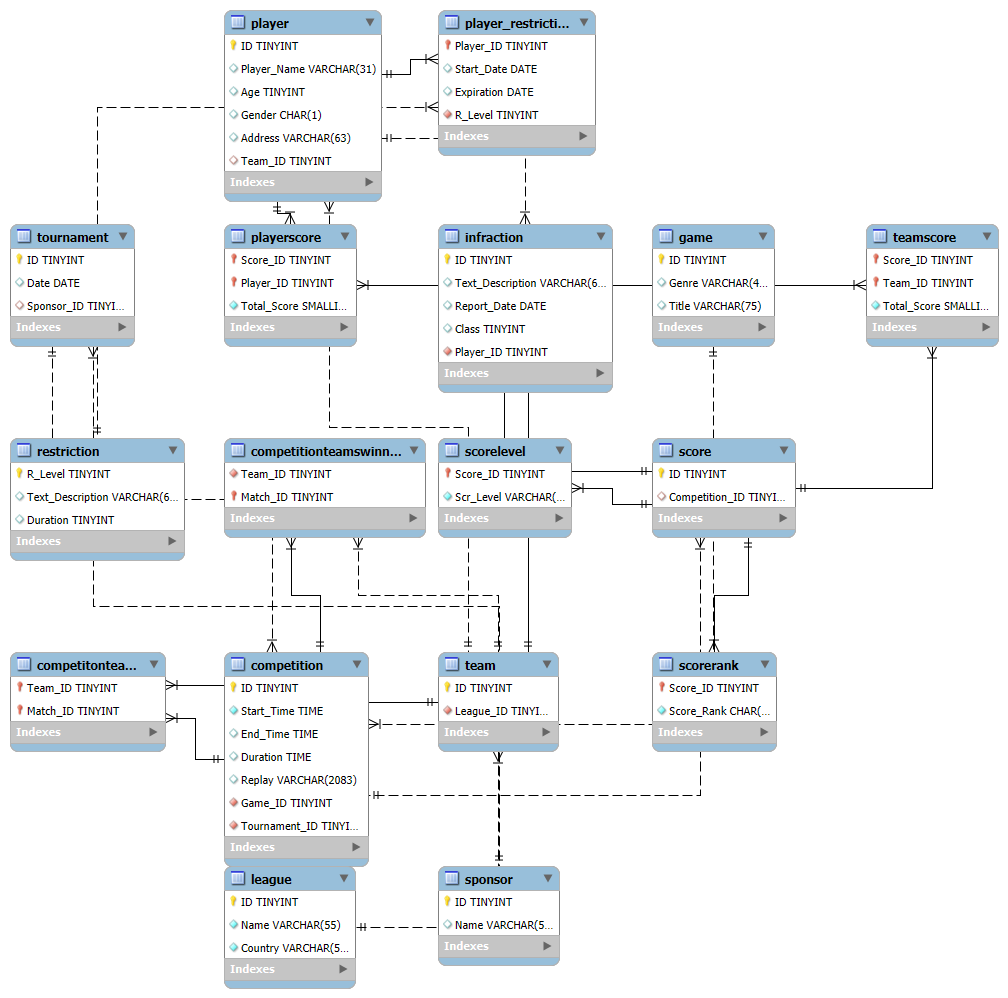
Diagram description:

The conceptual diagram contains 8 main entities: League, Team, Player, Score, Infraction, Game, Match, and Tournament. Teams are groups of associated players that play the same Game. Leagues are groups of Teams associated with the same organization. Matches are played in Tournaments, and each one is of a specific type of Game. Scores are recorded for Players and Teams for every Match. Any player Infractions are also recorded with associated restrictions

Entity-Relation Model Diagram

A diagram of a computer

AI-generated content may be incorrect.



**ER-Model Digram**

Diagram description:

Strong Entities:

League, Team, player, infraction, Restriction, Game, Sponsor, Tournament, Competition, Score.

Weak Entities:

Player\_Restriction, TeamScore, PlayerScore, CompetitionTeams, CompetitionTeamsWinner, ScoreLevel, and ScoreRank.

**Strong Entities Tables:**

* Team Table:
  + ID (tinyint unsigned, NOT NULL, AUTO\_INCREMENT) - Unique identifier
  + League\_ID(unsigned, NOT NULL) – References

▪ Store the information about team participants in the league.

▪ Team link to the Leagues (League\_ID to League)

* League Table:
  + ID (tinyint unsigned, NOT NULL, AUTO\_INCREMENT) - Unique identifier
  + Name (VARCHAR, 55, NOT NULL)
  + Country (VARCHR, 55, NOT NULL)

▪ Represent esport leagues having a multiple team

* Player Table:
  + ID (tinyint unsigned, NOT NULL, AUTO\_INCREMENT) - Unique identifier
  + Player\_Name (VARCHAR, 31)
  + Age (TINYINT)
  + Gender (CHAR, 1)
  + Adress (VRCHAR, 63)
  + Team\_ID(tinyint, unsigned)

▪ Players belong to teams (Team\_ID link to team)

▪Stored the personal and team-related information of each player.

* Infraction Table:
  + ID (tinyint unsigned, NOT NULL, AUTO\_INCREMENT) - Unique identifier
  + Text\_Description (VARCHAR, 63)
  + Report\_Date (Date)
  + Class (TINYINT)
  + Player\_ID - References

▪ Logs infraction or violations committed by players.

* Score Table:
  + ID (TINYINT, UNSIGNED, AUTO\_INCREMENT)) - Unique identifier
  + Competition\_ID (TINYINT, UNSIGNED) -References

▪ Store score information during the esport matches event associated with competions.

* Restriction Table:
  + R\_Level (TINYINT, NOT NULL, AUTO\_INCREMENT) -Restriction level identifier
  + Text\_Description (VARCHAR, 63)
  + Duration (tinyint, unsigned)

▪ Level the various types of restriction for players.

* Game Table:
  + ID (tinyint, Unsigned, Not Null) – Unique identifier
  + Genre (Varchar, 45)
  + Title (Varchar, 75)

▪ Information used in the competition

* Sponsor Table:
  + ID (tinyint, Unsigned, Not Null) – Unique Identifier
  + Name (VarChar, 50)

▪ Sponsors involvement in tournaments.

* Tournaments Table:
  + ID (tinyint, Unsigned, Not Null) – Unique Identifier
  + Date (Date)
  + Sponsor\_ID (tinyint unsigned)

▪Links tournament to sponsors.

▪Recorded tournaments events and their sponsors.

* Competition Table:
  + ID (tinyint Unsigned, Not Null) – unique identifier
  + Start\_Time (Time, Not Null)
  + End\_Time (Time)
  + Duration (Time)
  + Replay (VarChar, 2083)
  + Game\_ID (tinyint unsigned, not null) Game associated with the competition
  + Tournament\_ID (tinyint unsigned, not null)

▪ Track the individual matches within the tournament

**Weak Entities Tables:**

Depending on the existence of Table having it owns primary key.

Player\_Restriction: player (Player\_ID) and restriction (R\_Level)

▪Link the player with Various level of restriction types and durations

TeamScore: Score (Score\_ID) and Player (Player\_ID)

▪Record the team data got from the competition.

CompetitionTeams: Team (Team\_ID) and Competition (Match\_ID)

▪ Link the Team with the Matches

CompetitionTeamsWinner:Competiotion(Match) and references Team

▪ Track the Winning Team of each Competition,

ScoreLevel and ScoreRank: Score

▪Assign the level and the Rank (A, B, C, D…)

The approach used to develop your proposed solution

Team Contribution, planning and with strategies:

* Drafted the list of the projects and outlined the structure of the project.
* Selected the best project among them “Esport Database League”.
* Timeline frame of the project (Objectives, Task, and goals)
* Started with Conceptual and ER-Model Diagram using the tool Lucid Chart
* Build the Structure of the Tables as the project requirement with the help of Models diagram on My SQL.
* Implement the queries data and populate the tables.

Current releases

The current release is “ESports League Management v1.0”   
  
Key Features:

* Track Multiple Leagues
* Track Player Infractions
* Keep Scores and Ranks for Players and Teams
* Competition History, including match ups and winners

ESports League Management v1.0 provides data tracking features for one or more leagues. Within each league, team records are created. Likewise, within each team player records are created.   
  
As the league progresses, tournaments and competitions can be recorded. Records include the game played, teams involved, durations, and scores (individual and team). Additionally, violations may be tracked for players in the form of infractions and restrictions. Restrictions include age-based restrictions, temporary bans, and permanent bans.

Future releases

New Features

* Regional boundaries
  + Tracking team locations within Leagues to allow regional tournaments.
* Teams may participate in multiple Leagues
  + Functionality for Teams to participate in multiple Leagues allows for diverse aggregate Leagues.
* Inter-League tournaments
  + Inter-League tournaments introduce League Scores and competition tracking between Leagues.

Improvements

* User-friendly application interface
  + A user-friendly GUI makes the management software more accessible to ESports managers.

Releases

* New Features will be released as updates to version 1.0 in the form of version 1.x
* Major Improvements will be released as fully updated new versions x.0
  + Ex. GUI Release = “ESports League Management v2.0”

**Conclusion:**

In conclusion, our Project “Esports League Management Database” successfully built and implemented to manage leagues, teams, players, games, tournaments and scores. We started this project with Track every table Strong Entities and Weak Entities maintain relationships between them, ensuring data consistency, integrity, and efficient monitoring of esport activities. This project provides the strong foundation for managing competitions, recording the player performance, tracking the infraction, and create the opportunity for organizations, players, and sponsors involvement, To make the Valuable tool for the future growth of esports management.

Appendix

* Populated tables:

CREATE TABLE League(

ID tinyint unsigned NOT NULL AUTO\_INCREMENT,

Name VARCHAR(55) NOT NULL,

Country VARCHAR(55) NOT NULL,

PRIMARY KEY (ID)

);

CREATE TABLE Team(

ID tinyint UNSIGNED NOT NULL AUTO\_INCREMENT,

League\_ID tinyint UNSIGNED NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (League\_ID)

REFERENCES League(ID)

ON UPDATE CASCADE

ON DELETE CASCADE

);

CREATE TABLE Player(

ID tinyint unsigned NOT NULL AUTO\_INCREMENT,

Player\_Name VARCHAR(31),

Age TINYINT,

Gender CHAR(1),

Address VARCHAR(63),

Team\_ID tinyint unsigned,

PRIMARY KEY(ID),

FOREIGN KEY(Team\_ID) REFERENCES Team(ID)

ON UPDATE CASCADE

ON DELETE SET NULL,

CHECK (Gender IN ('M', 'F', 'N'))

);

CREATE TABLE Infraction(

ID tinyint unsigned NOT NULL AUTO\_INCREMENT,

Text\_Description VARCHAR(63),

Report\_Date DATE,

Class TINYINT,

Player\_ID tinyint unsigned NOT NULL,

PRIMARY KEY (ID),

FOREIGN KEY (Player\_ID) REFERENCES Player(ID)

ON UPDATE CASCADE

ON DELETE CASCADE

);

CREATE TABLE Restriction(

R\_Level TINYINT NOT NULL AUTO\_INCREMENT,

Text\_Description VARCHAR(63),

Duration tinyint unsigned,

PRIMARY KEY (R\_Level)

);

CREATE TABLE Player\_Restriction(

Player\_ID tinyint unsigned NOT NULL,

Start\_Date DATE,

Expiration DATE,

R\_Level TINYINT NOT NULL,

PRIMARY KEY(Player\_ID),

FOREIGN KEY(Player\_ID) REFERENCES Player(ID)

ON UPDATE CASCADE

ON DELETE CASCADE,

FOREIGN KEY(R\_Level) REFERENCES Restriction(R\_Level)

ON UPDATE CASCADE

ON DELETE CASCADE

);

CREATE TABLE Game(

ID tinyint Unsigned Not Null,

Genre Varchar(45),

Title Varchar(75),

Primary Key(ID)

);

CREATE TABLE Sponsor(

ID tinyint Unsigned Not Null,

Name VarChar(50),

Primary Key(ID)

);

CREATE TABLE Tournament(

ID tinyint Unsigned Not Null,

Date Date,

Sponsor\_ID tinyint unsigned,

Primary Key(ID),

Foreign Key(Sponsor\_ID)

References Sponsor(ID)

On Update cascade

);

CREATE TABLE Competition(

ID tinyint Unsigned Not Null,

Start\_Time Time Not Null,

End\_Time Time,

Duration Time,

Replay VarChar(2083),

Game\_ID tinyint unsigned not null,

Tournament\_ID tinyint unsigned not null,

Primary Key(ID),

foreign key(Game\_ID)

References Game(ID)

On Update Cascade,

foreign Key(Tournament\_ID)

References Tournament(ID)

On Update Cascade

);

-- Set names to be the same as "Competition" - Update if we change "Competition" to another name

CREATE TABLE Score (

ID TINYINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

Competition\_ID TINYINT UNSIGNED,

FOREIGN KEY (Competition\_ID) REFERENCES Competition (ID)

ON UPDATE CASCADE

ON DELETE CASCADE

);

CREATE TABLE ScoreLevel (

Score\_ID TINYINT UNSIGNED,

FOREIGN KEY (Score\_ID) REFERENCES Score (ID)

ON UPDATE CASCADE

ON DELETE CASCADE,

Scr\_Level VARCHAR(8) NOT NULL,

PRIMARY KEY (Score\_ID)

);

CREATE TABLE ScoreRank (

Score\_ID TINYINT UNSIGNED,

FOREIGN KEY (Score\_ID) REFERENCES Score (ID)

ON UPDATE CASCADE

ON DELETE CASCADE,

Score\_Rank CHAR(1) NOT NULL,

PRIMARY KEY (Score\_ID)

);

CREATE TABLE TeamScore (

Score\_ID TINYINT UNSIGNED,

Team\_ID TINYINT UNSIGNED,

FOREIGN KEY (Score\_ID) REFERENCES Score (ID)

ON UPDATE CASCADE

ON DELETE CASCADE,

FOREIGN KEY (Team\_ID) REFERENCES Team (ID)

ON UPDATE CASCADE

ON DELETE CASCADE,

Total\_Score smallint UNSIGNED NOT NULL,

PRIMARY KEY (Score\_ID, Team\_ID)

);

CREATE TABLE PlayerScore (

Score\_ID TINYINT UNSIGNED,

Player\_ID TINYINT UNSIGNED,

FOREIGN KEY (Score\_ID) REFERENCES Score (ID)

ON UPDATE CASCADE

ON DELETE CASCADE,

FOREIGN KEY (Player\_ID) REFERENCES Player (ID)

ON UPDATE CASCADE

ON DELETE CASCADE,

Total\_Score smallint UNSIGNED NOT NULL,

PRIMARY KEY (Score\_ID, Player\_ID)

);

CREATE TABLE CompetitionTeams(

Team\_ID tinyint Unsigned Not Null,

Match\_ID tinyint Unsigned Not Null,

Primary Key(Team\_ID, Match\_ID),

Foreign Key(Team\_ID) References Team(ID)

On Update Cascade,

Foreign Key(Match\_ID) References Competition(ID)

On Update Cascade

);

CREATE TABLE CompetitionTeamsWinner(

Team\_ID tinyint Unsigned Not Null,

Match\_ID tinyint Unsigned Not Null,

Primary Key(Match\_ID),

Foreign Key(Team\_ID) References Team(ID)

On Update Cascade,

Foreign Key(Match\_ID) References Competition(ID)

On Update Cascade

);

Database Samples:

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

* **SQL Scripts Used**
  + Data types
  + Tables created

-- PLAYER, INFRACTION, RESTRICTION, PLAYER\_RESTRICTION TEST QUERIES

-- number of players of each gender.

SELECT count(ID) as Player\_Count, Gender

FROM player

GROUP BY Gender;

A screenshot of a computer

AI-generated content may be incorrect.

-- select players whose infractions caused a restriction

SELECT player.ID, player.Player\_Name, infraction.Text\_Description as Infraction, infraction.Class as "Severity(1-10)", restriction.Text\_Description as Restriction, player\_restriction.Expiration

FROM player

INNER JOIN infraction ON infraction.Player\_ID = player.ID

INNER JOIN player\_restriction ON player.ID = player\_restriction.Player\_ID

INNER JOIN restriction ON player\_restriction.R\_Level = restriction.R\_Level;

A black screen with white text

AI-generated content may be incorrect.

-- select teams with at least two eligible players

SELECT player.Team\_ID

FROM player

LEFT JOIN player\_restriction ON player.ID = player\_restriction.Player\_ID

WHERE player\_restriction.R\_Level IS NULL OR player\_restriction.R\_Level = 2

GROUP BY Team\_ID

HAVING COUNT(player.ID) >= 2;

A black background with white numbers

AI-generated content may be incorrect.

-- select teams with no minors for 18+ tournament

SELECT player.Team\_ID

FROM player

LEFT JOIN player\_restriction ON player.ID = player\_restriction.Player\_ID

WHERE player\_restriction.R\_Level IS NULL OR player\_restriction.R\_Level != 2

GROUP BY Team\_ID

HAVING COUNT(player.ID) >= 2;

A black background with white numbers

AI-generated content may be incorrect.

-- SCORE QUERIES

-- How many scores of each grade by player and team

SELECT sl.Scr\_Level, sr.Score\_Rank, COUNT(\*)

FROM Score s

INNER JOIN ScoreLevel sl ON sl.Score\_ID = s.ID

INNER JOIN ScoreRank sr ON sr.Score\_ID = s.ID

GROUP BY sl.Scr\_Level, sr.Score\_Rank

ORDER BY sl.Scr\_Level DESC, sr.Score\_Rank ASC;

A screenshot of a computer program

AI-generated content may be incorrect.

-- Top ten player scores

SELECT p.Player\_Name, g.Title, ps.Total\_Score, sr.Score\_Rank

FROM PlayerScore ps

INNER JOIN Score s ON ps.Score\_ID = s.ID

INNER JOIN ScoreRank sr ON sr.Score\_ID = s.ID

INNER JOIN Competition c ON c.ID = s.Competition\_ID

INNER JOIN Game g ON g.ID = c.Game\_ID

INNER JOIN Player p ON p.ID = ps.Player\_ID

ORDER BY ps.Total\_Score DESC

LIMIT 10;

A screenshot of a computer screen

AI-generated content may be incorrect.

-- Top ten team scores

SELECT t.ID AS Team\_ID, g.Title, ts.Total\_Score, sr.Score\_Rank

FROM TeamScore ts

INNER JOIN Score s ON ts.Score\_ID = s.ID

INNER JOIN ScoreRank sr ON sr.Score\_ID = s.ID

INNER JOIN Competition c ON c.ID = s.Competition\_ID

INNER JOIN Game g ON g.ID = c.Game\_ID

INNER JOIN Team t ON t.ID = ts.Team\_ID

ORDER BY ts.Total\_Score DESC

LIMIT 10;

A screenshot of a computer screen

AI-generated content may be incorrect.

-- Average player performance by game

SELECT g.Title, ROUND(AVG(ps.Total\_Score), 1) AS Performance

FROM PlayerScore ps

INNER JOIN Score s ON s.ID = ps.Score\_ID

INNER JOIN Competition c ON c.ID = s.Competition\_ID

INNER JOIN Game g ON g.ID = c.Game\_ID

GROUP BY g.Title

ORDER BY Performance DESC;

A screen shot of a computer

AI-generated content may be incorrect.

-- Average team performance by game

SELECT g.Title, ROUND(AVG(ts.Total\_Score), 1) AS Performance

FROM TeamScore ts

INNER JOIN Score s ON s.ID = ts.Score\_ID

INNER JOIN Competition c ON c.ID = s.Competition\_ID

INNER JOIN Game g ON g.ID = c.Game\_ID

GROUP BY g.Title

ORDER BY Performance DESC;

A screen shot of a computer

AI-generated content may be incorrect.

SELECT \* FROM CompetitionTeamsWinner;

A screenshot of a computer screen

AI-generated content may be incorrect.

* **Time Log of Each Members**