# Table of Contents

# 1. Project Milestone 1

## 1.1 Application Description

The sport platform wants to have a new database to store information about sport events. The platform stores. Sport events can be hosted in-person, virtual or both. The platform wants to store information about the players, organizers, viewers, the team the players are in and the sport. The team only records the current roster and players can only be assigned to at most 1 team at any given time. Details about the Organization are stored in the organizers. The Sport Event is created when it is organized by an organizer.

Explanation of relationships:
Team - Plays - Sport: 1 team can play 1 and only 1 sport; A sport will have at least 1 team, to many teams that play it.
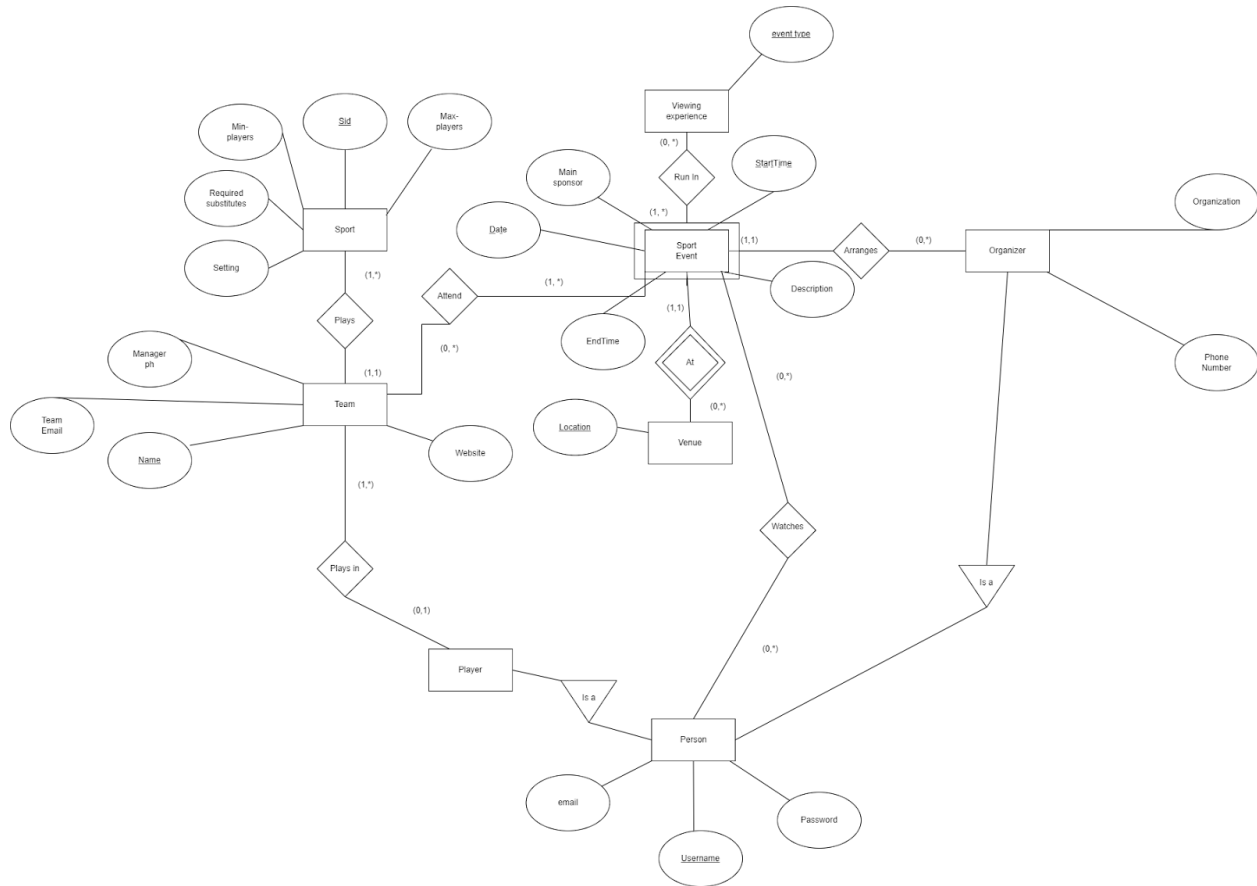Sport - Sport - Event: A sport can take place at no events, to many events; A sport event can hold one to many sports
Viewer - Attends - Sport Event: 1 viewer can attend 0 events or many events; A sport event can have 0 to many members in attendance
Sport-event - arranges - Organizer: A sport event is organized by one organizer and only 1, An organizer can organize 0 to many events.

A sporting event is a generalization between virtual esports and other forms of sports that do not have to be played from a shared location.

## 1.2 Revised ER Diagram

# 2. Project Milestone 2

## 2.1 Relation Schema

**Person**
Username, Email, Password, First Name, Last Name
**Player (Inherit from Person)**
Username, TeamID
**Organizer (Inherit from Person)**
Username, Organization, Phone Number
**Team**
TeamID, Name, TeamEmail, Phone Number, Website, Sport
**Sport**
Name, MinPlayer, MaxPlayer, Required Substitute, Setting
**Setting**
Setting
**Sport Event**
Date, Start Time, Location, End Time, Main Sponsor, Description, Organizer
**Venue**
Location
**Viewing experience**
Event-type
**Run In**
Date, Start Time, Organizer, Event-type
**Attends**
TeamID, Date, Start Time, Location
**Watches**
Person, Date, Start Time, Location

## 2.2 Table Definitions

```
create table Person(
Username varchar(30),
Email varchar(50) not null,
Password varchar(30) not null,
FirstName varchar(30) not null,
LastName varchar(30) not null,

CONSTRAINT PKPerson PRIMARY KEY (Username),
CONSTRAINT emailCheck check (email LIKE '%_@_%.___%')
);
```

```sql
create table Organizer(
Username varchar(30),
Organization varchar(50) not null,
PhoneNumber varchar(20),

CONSTRAINT PKOrg PRIMARY KEY (Username),
CONSTRAINT FKOrg FOREIGN KEY (Username) REFERENCES Person ON DELETE CASCADE
);


create table Player(
Username varchar(30),
TeamID varchar(30),

CONSTRAINT PKPlayer PRIMARY KEY (Username),
CONSTRAINT FKPlayer FOREIGN KEY (Username) REFERENCES Person ON DELETE CASCADE,
CONSTRAINT FKTeamID FOREIGN KEY (TeamID) REFERENCES Team
);
create table Setting(
SettingID varchar(30),

CONSTRAINT PKSettingID PRIMARY KEY (SettingID)
);


create table Setting(
SettingID varchar(30),

CONSTRAINT PKSettingID PRIMARY KEY (SettingID)
);


create table Sport(
Name varchar(30),
MinPlayer integer not null,
MaxPlayer integer not null,
RequiredSubstitute varchar(50),
SettingID varchar(30),

CONSTRAINT PKSport PRIMARY KEY (Name),
CONSTRAINT FKSetting foreign key(SettingID) references Setting(SettingID)
);
```

```sql
create table Team(
 TeamID varchar(30),
 Name varchar(30) not null,
 TeamEmail varchar(50) not null,
 PhoneNumber varchar(20),
 Website varchar(50) not null,
 Sport varchar(30),

 CONSTRAINT PKTeam PRIMARY KEY (TeamID),
 CONSTRAINT FKSport foreign key(Sport) references Sport(Name)
);

create table Venue(
 Location varchar(30),

 CONSTRAINT PKLocation PRIMARY KEY (Location)
);

create table SportEvent(
 EventDate Date,
 StartTime Date,
 Location varchar(30),
 EndTime Date,
 Organizer varchar(30),
 MainSponsor varchar(50) not null,
 Description varchar(100),

 CONSTRAINT PKSportEvent PRIMARY KEY (EventDate, StartTime, Location),
 CONSTRAINT FKLocation FOREIGN KEY(Location) REFERENCES Venue(Location),
 CONSTRAINT FKOrganizer FOREIGN KEY(Organizer) REFERENCES Organizer(Username)
);
```

S

```sql
create table ViewingExperience(
 EventType varchar(30),

 CONSTRAINT PKViewingExperience PRIMARY KEY (EventType)
);

create table RunIn(
 EventDate date,
 StartTime date,
 Location varchar(30),
 EventType varchar(30),

 CONSTRAINT PKIn PRIMARY KEY (EventDate, StartTime, Location, EventType),
 CONSTRAINT FKEventDate foreign key(EventDate, StartTime, Location) references SportEvent(EventDate, StartTime, Location),
 CONSTRAINT FKEventType foreign key(EventType) references ViewingExperience(EventType)
);
```

```sql
create table RunIn(
EventDate date,
StartTime date,
Location varchar(30),
EventType varchar(30),

CONSTRAINT PKIn PRIMARY KEY (EventDate, StartTime, Location, EventType),
CONSTRAINT FKEventDate foreign key(EventDate, StartTime, Location) references SportEvent(EventDate, StartTime, Location),
CONSTRAINT FKEventType foreign key(EventType) references ViewingExperience(EventType)
);

create table Attends(
TeamID varchar(30),
EventDate date,
StartTime date,
Location varchar(30),

CONSTRAINT PKAttends PRIMARY KEY (TeamID, EventDate, StartTime, Location),
CONSTRAINT FKTeam foreign key(TeamID) references Team(TeamID),
CONSTRAINT FKEventDate2 foreign key(EventDate, StartTime, Location) references SportEvent(EventDate, StartTime, Location)
);

create table Watches(
Person varchar(30),
EventDate date,
StartTime date,
Location varchar(30),

CONSTRAINT PKWatches PRIMARY KEY (Person, EventDate, StartTime, Location),
CONSTRAINT FKPerson foreign key(Person) references Person(Username),
CONSTRAINT FKEventDate3 foreign key(EventDate, StartTime, Location) references SportEvent(EventDate, StartTime, Location)
);
```

## 2.3  Dataset
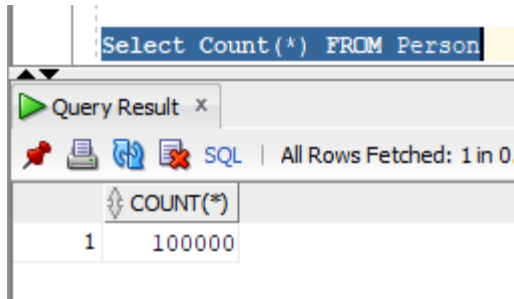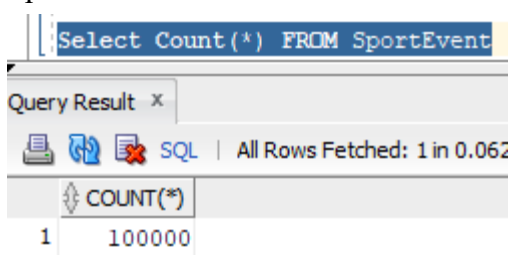
### 2.3.1  Small


### 2.3.2  Large

Person Table



Sport Event Tables



The screenshot above are the two main table.
s
The dataset is randomly generated using a java program generating random string of chars/integer.

## 2.4  Application

# 3. Project Milestone 3

## 3.1 Indexing and Querying Optimization

### 3.1.1 Queries and SQL script to create index

```sql
Create Index btree on Person(firstname);

 Select * from Person where firstname like 'nu%q%';
create table SportEvent(
EventDate Date,
StartTime Date,
Location varchar(30),
EndTime Date,
Organizer varchar(30),
MainSponsor varchar(50) not null,
Description varchar(100),

CONSTRAINT PKSportEvent PRIMARY KEY (EventDate, StartTime, Location),
CONSTRAINT FKLocation FOREIGN KEY(Location) REFERENCES Venue(Location),
CONSTRAINT FKOrganizer FOREIGN KEY(Organizer) REFERENCES Organizer(Username)
)cluster hashCluster(Organizer);

Create cluster hashCluster (
    Organizer varchar(30))
    size 256 HASHKeys 100;

 Select * from SportEvent where organizer = 'umtbverr90633';
```

### 3.1.2 Why was it chosen?

### 3.1.3 Performance Measurement

B-Tree Index Before Implement

```
 1 Plan hash value: 1493655343
 2
 3 -------------------------------------------------------------------
 4 | Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
 5 -------------------------------------------------------------------
 6 |   0 | SELECT STATEMENT   |         |   124 |  8556 |   308   (1)| 00:00:01 |
 7 |*  1 |   TABLE ACCESS FULL| PERSON  |   124 |  8556 |   308   (1)| 00:00:01 |
 8 -------------------------------------------------------------------
 9
10 Predicate Information (identified by operation id):
11 ---------------------------------------------------
12
13    1 - filter("FIRSTNAME" LIKE 'nu%q%')
```

B-Tree Index After Implement

```
 1 Plan hash value: 4105735724
 2
 3 -----------------------------------------------------------------------------------------
 4 | Id  | Operation                           | Name   | Rows  | Bytes | Cost (%CPU)| Time     |
 5 -----------------------------------------------------------------------------------------
 6 |   0 | SELECT STATEMENT                    |        |   124 |  8556 |   126   (0)| 00:00:01 |
 7 |   1 |   TABLE ACCESS BY INDEX ROWID BATCHED| PERSON |   124 |  8556 |   126   (0)| 00:00:01 |
 8 |*  2 |    INDEX RANGE SCAN                 | BTREE  |   124 |       |     2   (0)| 00:00:01 |
 9 -----------------------------------------------------------------------------------------
10
11 Predicate Information (identified by operation id):
12 ---------------------------------------------------
13
14    2 - access("FIRSTNAME" LIKE 'nu%q%')
15        filter("FIRSTNAME" LIKE 'nu%q%')
```

## Hash-Cluster Index Before Implement

```
 1 Plan hash value: 1484609831
 2
 3 ---------------------------------------------------------------------
 4 | Id  | Operation          | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
 5 ---------------------------------------------------------------------
 6 |   0 | SELECT STATEMENT   |            |  1000 | 74000 |    308   (1)| 00:00:01 |
 7 |*  1 |  TABLE ACCESS FULL | SPORTEVENT |  1000 | 74000 |    308   (1)| 00:00:01 |
 8 ---------------------------------------------------------------------
 9
10 Predicate Information (identified by operation id):
11 ---------------------------------------------------
12
13    1 - filter("ORGANIZER"='umtbverr90633')
```

## Hash-Cluster Index After Implement

```
 1 Plan hash value: 4210168801
 2
 3 ---------------------------------------------------------------------
 4 | Id  | Operation          | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
 5 ---------------------------------------------------------------------
 6 |   0 | SELECT STATEMENT   |            |  1317 |  180K |      1   (0)| 00:00:01 |
 7 |*  1 |  TABLE ACCESS HASH | SPORTEVENT |  1317 |  180K |      1   (0)| 00:00:01 |
 8 ---------------------------------------------------------------------
 9
10 Predicate Information (identified by operation id):
11 ---------------------------------------------------
12
13    1 - access("ORGANIZER"='umtbverr90633')
14
15 Note
16 -----
17    - dynamic statistics used: dynamic sampling (level=2)
```

### 3.1.4  Discussion of Performance Measurement

# 4. COMPX323-22A Project Checklist

Kevin Han:                (Contributions %)
Bedir Asici:              (Contributions %)
Tetsusaburo Kato:      (Contributions %)

| | Project Milestone 1 | |
|---|---|---|
| 1 | 1a. Clear structure of milestone material, including headings, sections, readable screenshots with captions. This checklist should be included and filled in. | |
| 2 | 1b. Database application description. | |
| 6 | 1c. Revised ER Diagram. | ✔ |
| | **Project Milestone 2** | |
| 4 | 2. Relational schema for your ER Diagram. | |
| 6 | 3. Table definitions in Oracle, include SQL script which creates relevant tables etc. | |
| 3 | 4a. Dataset: small (screenshots of dataset successfully loaded). | |
| 4 | 4b. Dataset: large.<br>Description of how data was created (incl code if relevant)<br>Screenshot of large dataset successfully loaded (use count). | |
| | 5. Application:<br>Functionality to display and modify the database.<br>System should be error proof with appropriate user messages.<br>Screenshots showing functionality, with appropriate descriptions. | |
| | **Project Milestone 3** | |
| | 6. Indexing and Query Optimization:<br>Show queries used and SQL script that creates the indexes.<br>Discussion of why these indexes were chosen to optimize the queries.<br>Performance measurements with and without indexes (with query plan).<br>Discussion of performance measurements. | |
| | 7. Application: extend with MongoDB.<br>MongoDB version of database (show structure) + small dataset (screenshot).<br>Explanation of the data structures you have chosen and comparison to your SQL version.<br>Core functionality of application in second tab/area, using MongoDB (screenshots). | |