📖 Database Design Requirements

Assignment Requirements:

COS20031 Design

Naming Conventions:

We chose to have 2 different naming conventions to give clarity and ease of maintenance on the database design.

The naming convention for TABLES: snake_case

Requirements:

- Replace spaces with _ underscores
- No dots, underscores, numbers, dashes, or any other special characters are allowed within the word
- Start all names with letters, do not end with trailing _ underscores
  - **Example: dog_water**, **dog_food**, and **archer_score**
  - **NOT TO DO**: **_dogWater**, **dogFood_**, and **1_archerScore_**

The naming convention for FIELDS: camelCase

Requirements:

- No spaces
- No dots, underscores, numbers, dashes, or any other special characters are allowed within the word
- Start all names with lower case
  - **Example: dogWater**, **dogFood**, and **archerScore**
  - **NOT TO DO: dogwater, DOGFOOD,** and **ArcherScorE**

Diagram

Reasonings:

[Add your work on the database, the diagram and discussions and reasoning, to your Confluence space.]

The club_championship table is has been denormalised due to the more helpful arrangement of keeping clubChampionshipID and clubChampionshipName in the same table instead of creating another lookup table. Since this would have fewer entries due to only have linking club championships and competitions the extra clutter is reasonable. This calculation can be easily moved to another table.

Business Rules:

- The target is placed at a set distance of 20m, 30m, 40m, 50m, 60m, 70m, or 90m.
- There are two types of target faces: 80cm and 122cm.
- Archers shoot 6 arrows per end.
- A round can consist of one or more distances and have ranges of 5 or 6 ends.
- Rounds can have 2-4 ranges, but there are old rounds that are not limited to four.
- Archers are classified by age and gender into 16 categories.
- There are 5 legal types of equipment: Recurve, Compound, Recurve Barebow, Compound Barebow, and Longbow.

- Class and division define an archer's category.
- Archers compete within their own category.
- Some club competitions form part of the yearly club championship.

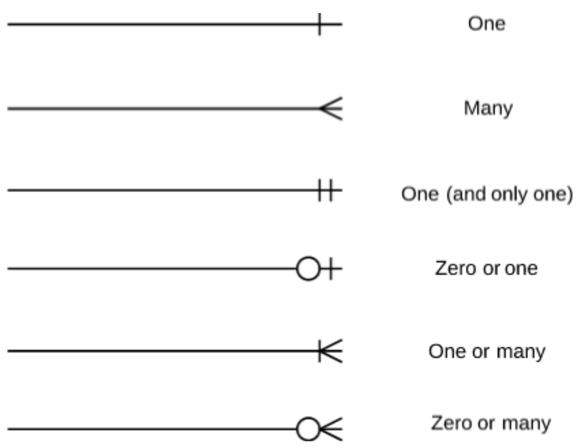
Constraints

- Each arrow score must be identifiable in terms of which end it belongs to.
- Each end must be identifiable in its position in the round score (range, end).
- Arrow scores within an end are recorded from highest to lowest, without tracking the order in which they were shot.
- Archers must enter their scores into a staging table using a hand-held device.
- Scores must contain arrow-by-arrow scores.
- The database must contain all information needed to identify an archer's division (age, gender).
- The equivalent rounds must be time-dependent, as Archery Australia sometimes changes them.

Assumptions:

- All competitions are held on the same day.
- Archers can look up their own scores over time, with the ability to restrict the number of scores by date range and type of round.
- Score listings can be sorted by date and score.
- Archers can record dates, times, rounds, and equipment for their scores.
- Archers can look up definitions of rounds and equivalent rounds.
- The database will show club competition results, including placements and scores.
- The participating rounds and scores for club championships are defined so that results can be shown and winners identified.
- Archers can look up their personal best (PB) score for a round, as well as the club's best score and the archer who shot it.
- The recorder can enter new archers, new rounds, and new competitions.
- The recorder can add new scores that archers have staged on their mobile devices.
- Some scores must be linkable to a competition, and some competitions must be identifiable as part of a club championship.
- There is a definition of the default equipment, allowing the identification of the category in the absence of user input.
- A history of equivalent rounds is maintained to ensure past competitions remain valid.

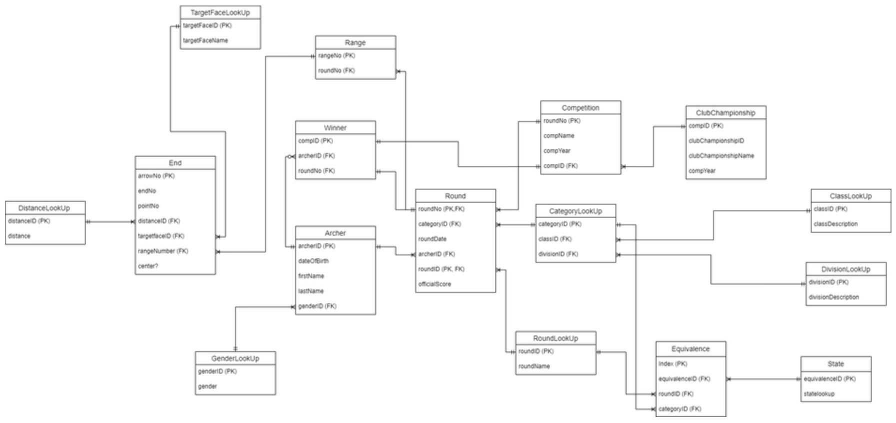
Resources:



Relationships labels

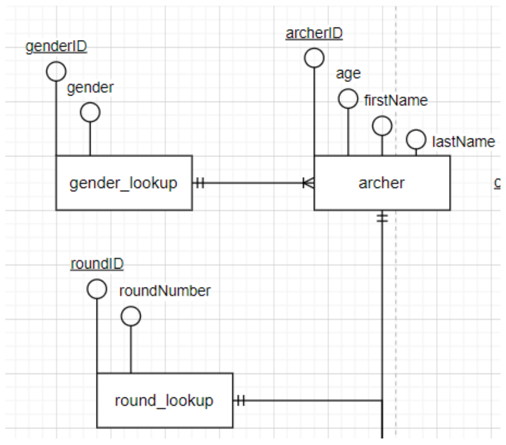
ER diagram (FINAL)

Here is the final version of ER diagram:

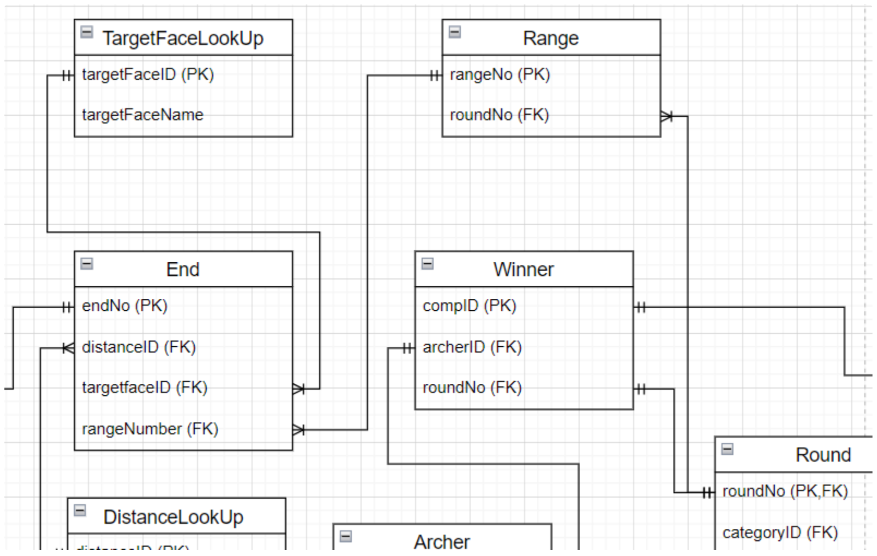


⭐ ERD note

The way the ERD drawn is changed from



to



This is because the first way of creating ERD does not show all the attributes for the entities (we have to take foreign key in the other table), so it can be very difficult for the people like students to understand the database. So we will use the second way of drawing ERD diagram, which show all the attibutes.

All the weak entities are converted to strong entities.

Drafts ERD

This dropdown consist of all the drafts we have before finalising the database.

ER diagram (First Draft)

ER Diagram (Second Draft)

ERD Diagram (Third Draft)

ERD diagram (Fourth Draft)

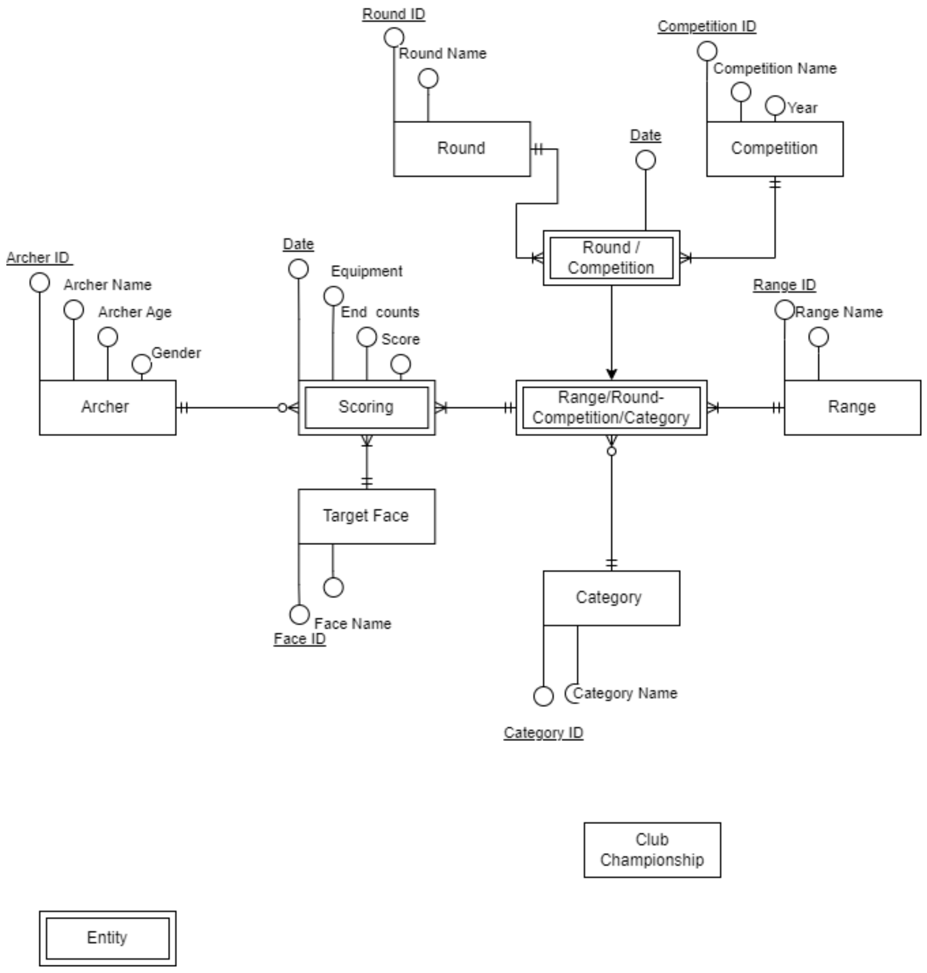
ERD diagram (Fifth Draft)

ERD (Sixth Draft)

ERD diagram (Backup and Alternative)

ER diagram (First Draft)

This is a draft ERD of our archey scoring database system (Club Championship is an entity we still need to work on)



The underlined attribute is the primary key of the entity

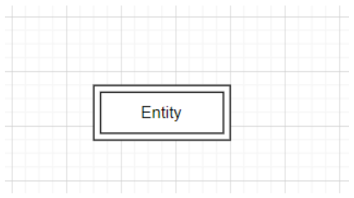


Figure 2: Weak entity

In the diagram I

🚩 ER Diagram -- Proposed changes

Range – table

-needs a distances lookup table

Category – table

-needs lookup table for classes (age-group-class)

-needs lookup table for divisions (bow-type)

Round – table

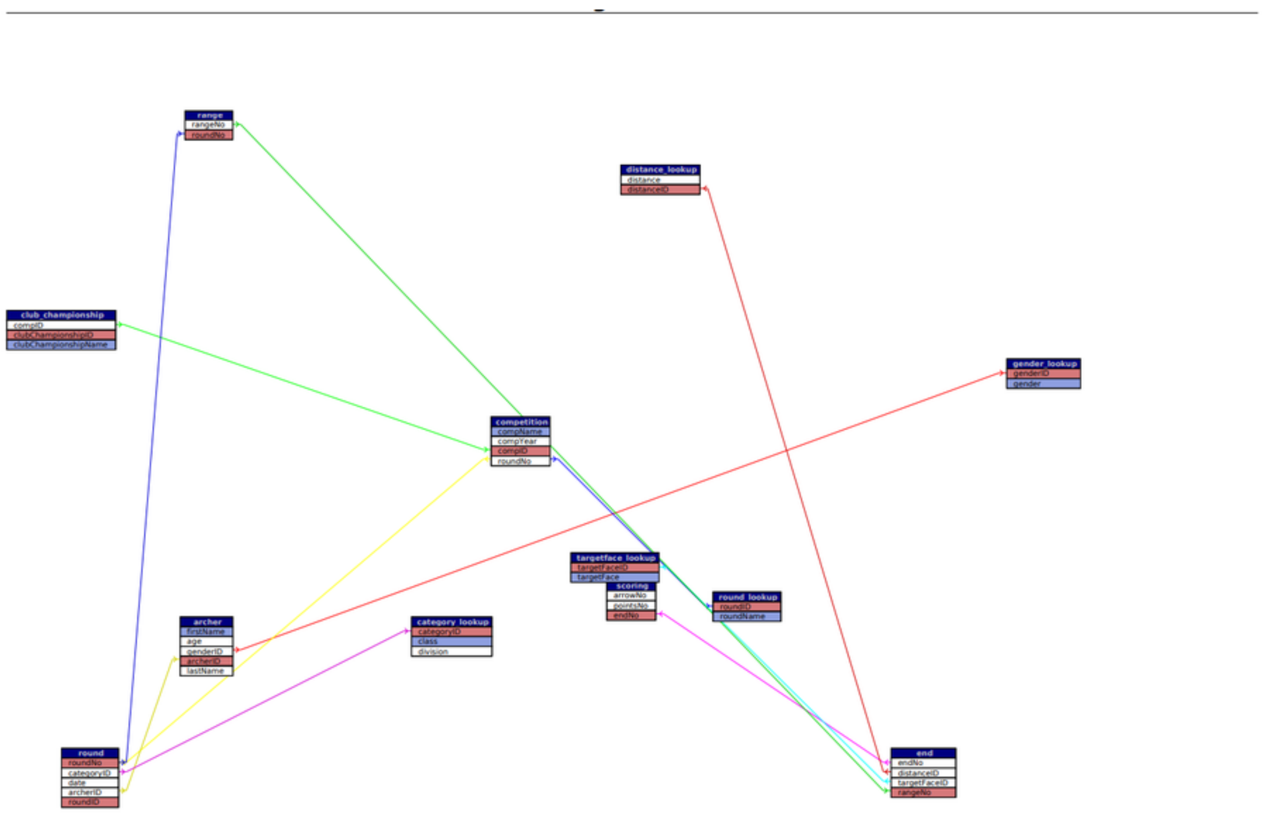
-needs a boolean for whether is useable as an equivalent round

Unresolved Issues

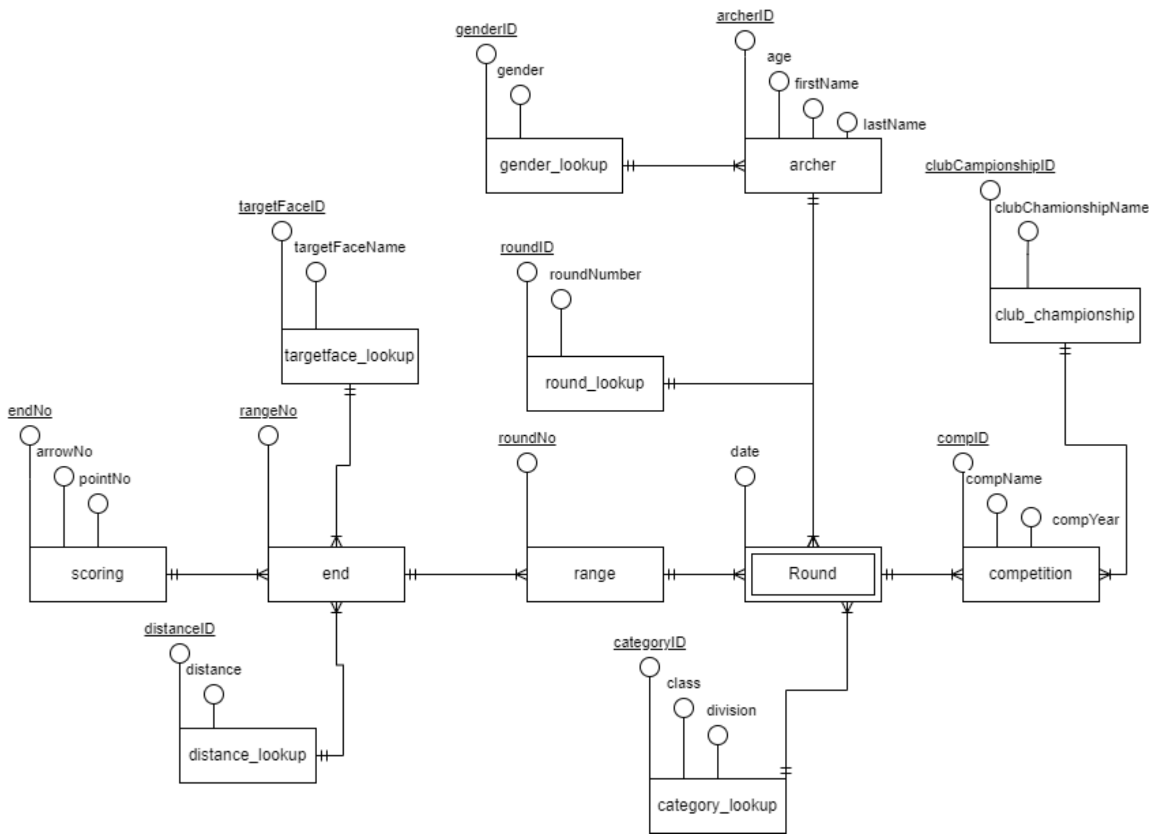
-if an archer shoots the same scores for an end on the same day

ER Diagram (Second Draft)

Here is the diagram in phpmyadmin database view



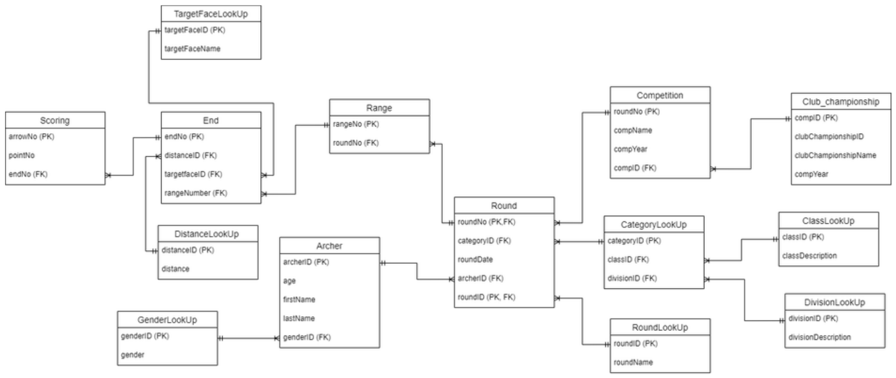
Here is the ERD diagram (second draft)



genderID
gender
gender_lookup
archerID
age
firstName
lastName
archer
clubCampionshipID
clubChamionshipName
club_championship
targetFaceID
targetFaceName
targetface_lookup
roundID
roundNumber
round_lookup
endNo
arrowNo
pointNo
scoring
rangeNo
end
roundNo
range
date
Round
compID
compName
compYear
competition
distanceID
distance
distance_lookup
categoryID
class
division
category_lookup

ERD Diagram (Third Draft)

Here is the ERD diagram after some addition and adjustment:

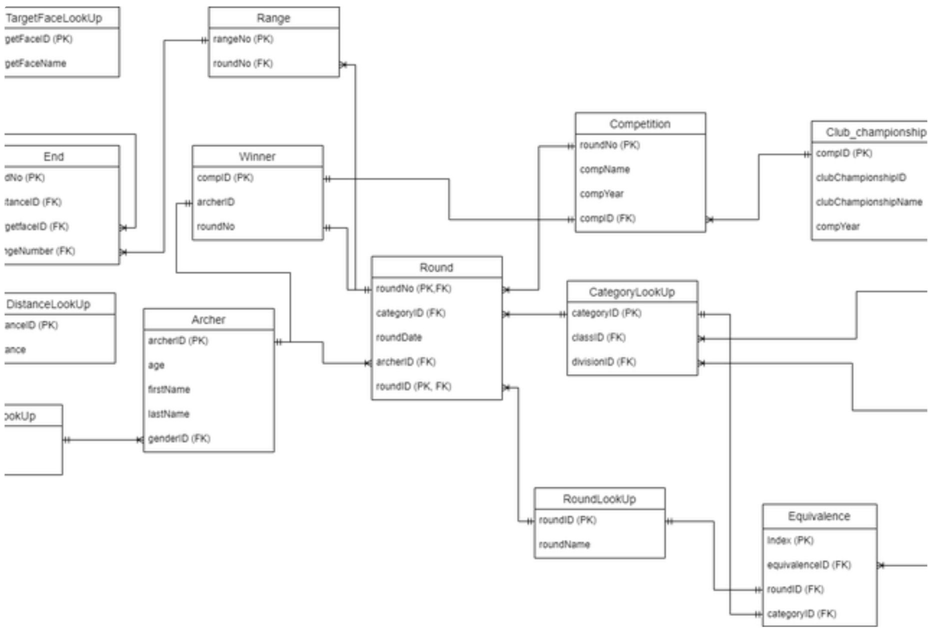


Here is the PDF containing the design of the database (exported from phpmyadmin)

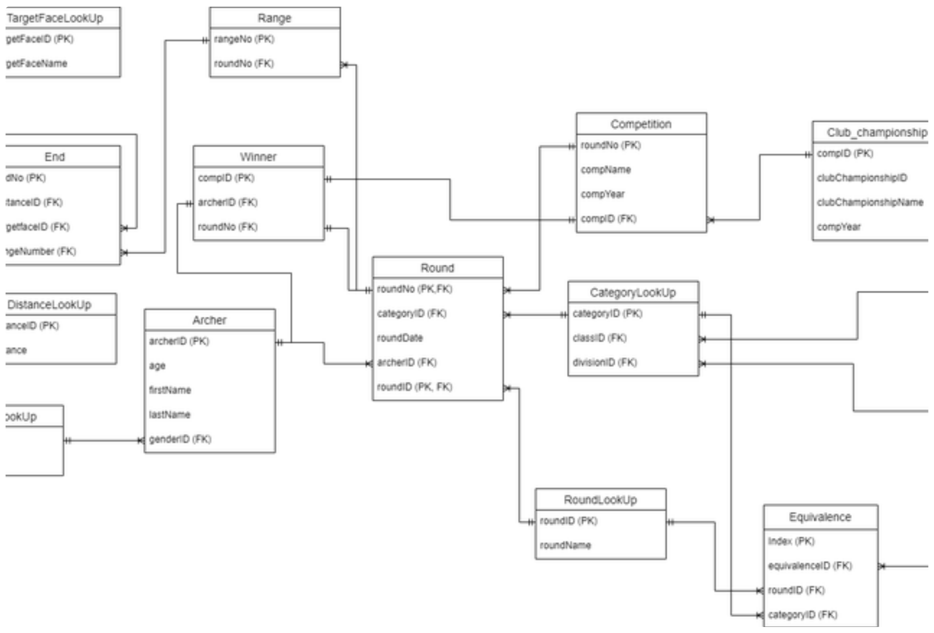
COS20031.pdf
20 Apr 2023, 03:26 pm

ERD diagram (Fourth Draft)

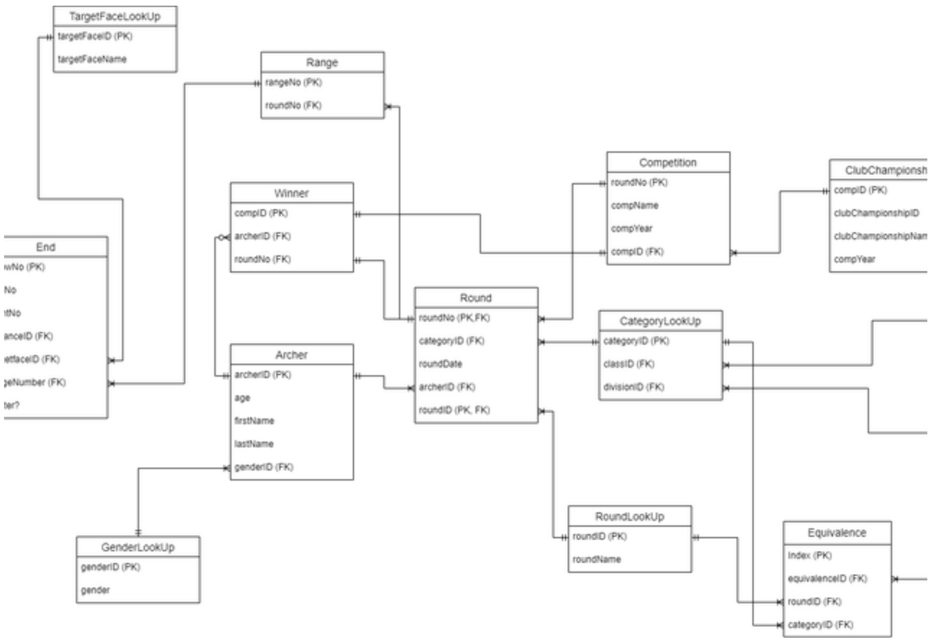
Here is the new ERD diagram after adjustment:



ERD diagram (Fifth Draft)

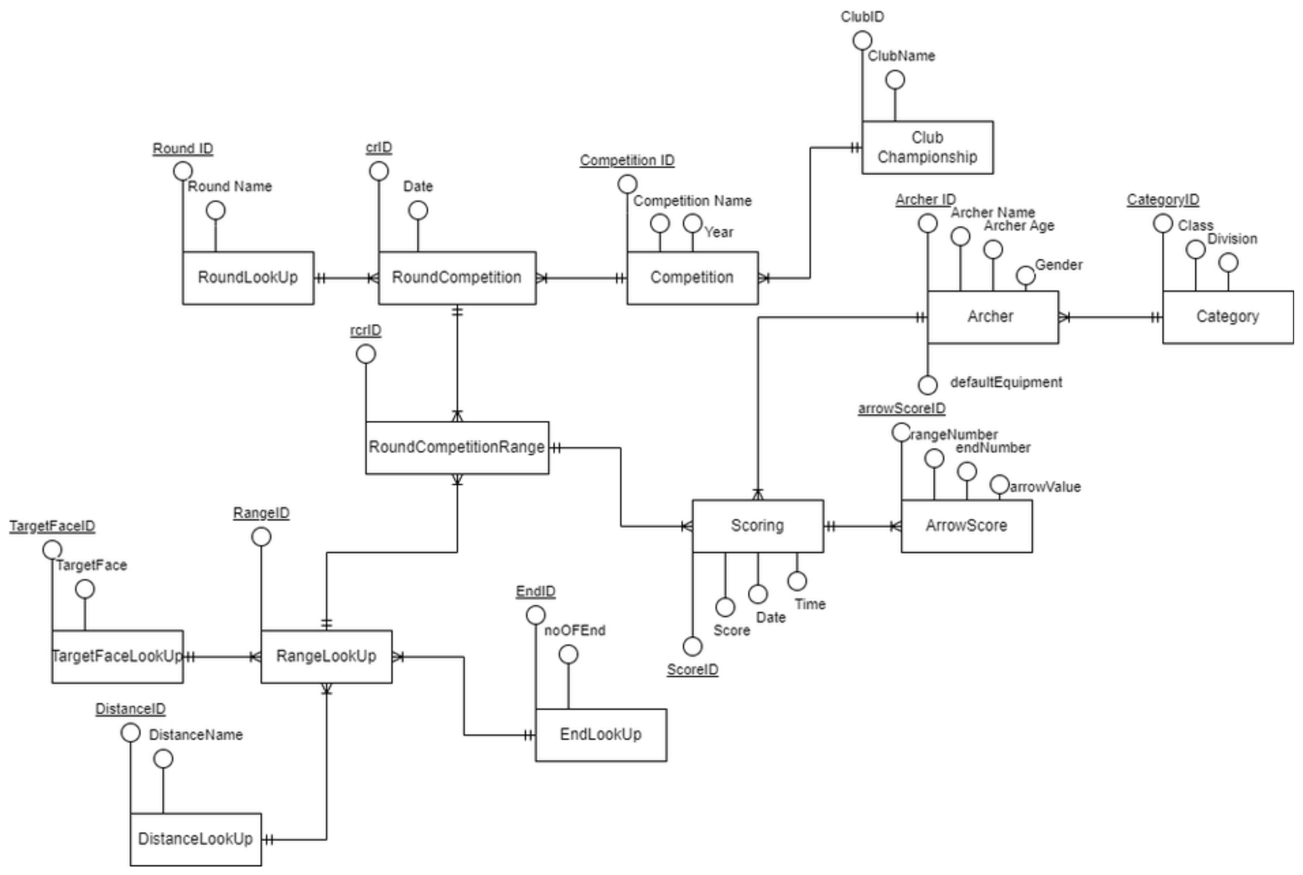


ERD (Sixth Draft)



ERD diagram (Backup and Alternative)

Here is the backup database:



📘 Database Normalisation (Week 5)

Review of ER diagram to normalise/denormalise, revise relationships (Week 5), possible subsequent reviews (e.g. adjustments due to use cases)

Overview

The database is in the Third Normal Form (3NF), as there are no transitive dependencies or any non-prime attributes depending on other non-prime attributes. This is important as it reduces redundancy and improves the integrity of data.

Normalisation process for the database:

1. **First Normal Form (1NF):** Each table has a primary key: uniquely identifying each record in a table. The values in each column of a table are atomic (indivisible). There are no repeating or multi-valued groups.

2. **Second Normal Form (2NF):** Each table in the database seems to observe 2NF, i.e., it's in 1NF and all non-key attributes are fully functionally dependent on the entire primary key.

3. **Third Normal Form (3NF):** The database is in 3NF, i.e., it's in 2NF and there are no transitive dependencies. Every non-prime attribute is non-transitively dependent on every key of the table.

Here's a detailed look at a few of the 3NF tables:

- `archer` table: This table is in 3NF. The primary key is `archerID`, and all other attributes (`firstName`, `dateOfBirth`, `genderID`, `lastName`) are dependent on it. There are no transitive dependencies.

- `gender_lookup` table: This lookup table is in 3NF. The primary key is `genderID` and the `gender` attribute is dependent on it.

- `category_lookup` table: This table is also in 3NF. The primary key is `categoryID`, and the `classID` and `divisionID` attributes are dependent on it.

- `class_lookup`, `division_lookup`, `distance_lookup`, `round_lookup`, `targetface_lookup` tables: These are lookup tables and are all in 3NF. They each have a primary key and a description attribute that depends on it.

Update

The competition and club_championship have been denormalised to 2NF, this is to allow for their ID numbers able to reference different parts of the same competition or club championship.

e.g. NYSE Day 1, NYSE Day 2

Since these fields are for context only their are not required to be limited to exact spelling, as the ID numbers are more important and easily confirmed for competition creation.

Physical database - Create Table statements (Week 6)

SQL File to create the COS20031 Database and its tables:

COS20031.sql
11 May 2023, 09:31 am

SQL Code:

```sql
-- phpMyAdmin SQL Dump
-- version 5.2.1
-- https://www.phpmyadmin.net/
--
-- Host: localhost
-- Generation Time: May 11, 2023 at 09:23 AM
-- Server version: 10.6.12-MariaDB
-- PHP Version: 7.2.34

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `COS20031`
--

-- --------------------------------------------------------

--
-- Table structure for table `archer`
--

CREATE TABLE `archer` (
  `firstName` varchar(50) NOT NULL,
  `dateOfBirth` date NOT NULL,
  `genderID` int(11) NOT NULL,
  `archerID` int(11) NOT NULL,
  `lastName` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `category_lookup`
--

```

```
CREATE TABLE `category_lookup` (
  `categoryID` int(11) NOT NULL,
  `classID` int(25) NOT NULL,
  `divisionID` int(25) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `class_lookup`
--

CREATE TABLE `class_lookup` (
  `classID` int(11) NOT NULL,
  `classDescription` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `club_championship`
--

CREATE TABLE `club_championship` (
  `compID` int(11) NOT NULL,
  `clubChampionshipID` int(11) NOT NULL,
  `clubChampionshipName` varchar(25) NOT NULL,
  `compYear` int(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `competition`
--

CREATE TABLE `competition` (
  `compName` varchar(25) NOT NULL,
  `compYear` int(11) NOT NULL,
  `compID` int(11) NOT NULL,
  `roundNo` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `distance_lookup`
--

CREATE TABLE `distance_lookup` (
  `distanceID` int(11) NOT NULL,
  `distance` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `division_lookup`
```

```sql
--

CREATE TABLE `division_lookup` (
  `divisionID` int(11) NOT NULL,
  `divisionDescription` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `end`
--

CREATE TABLE `end` (
  `arrowNo` int(11) NOT NULL,
  `endNo` int(11) NOT NULL,
  `pointsNo` int(11) NOT NULL,
  `center` tinyint(4) NOT NULL DEFAULT 0,
  `distanceID` int(11) NOT NULL,
  `targetFaceID` int(11) NOT NULL,
  `rangeNo` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `equivalence`
--

CREATE TABLE `equivalence` (
  `indexID` int(11) NOT NULL,
  `equivalenceID` int(11) NOT NULL,
  `roundID` int(11) NOT NULL,
  `categoryID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `gender_lookup`
--

CREATE TABLE `gender_lookup` (
  `genderID` int(11) NOT NULL,
  `gender` varchar(25) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `range`
--

CREATE TABLE `range` (
  `rangeNo` int(11) NOT NULL,
  `roundNo` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

```

```
-- --------------------------------------------------------

--
-- Table structure for table `round`
--

CREATE TABLE `round` (
  `roundNo` int(11) NOT NULL,
  `categoryID` int(11) NOT NULL,
  `date` date NOT NULL,
  `archerID` int(11) NOT NULL,
  `roundID` int(11) NOT NULL,
  `officialScore` tinyint(4) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `round_lookup`
--

CREATE TABLE `round_lookup` (
  `roundID` int(11) NOT NULL,
  `roundName` varchar(25) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `state`
--

CREATE TABLE `state` (
  `equivalenceID` int(11) NOT NULL,
  `state` tinyint(4) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `targetface_lookup`
--

CREATE TABLE `targetface_lookup` (
  `targetFaceID` int(11) NOT NULL,
  `targetFace` varchar(25) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
-- Table structure for table `winner`
--

CREATE TABLE `winner` (
  `roundNo` int(11) NOT NULL,
  `compID` int(11) NOT NULL,
  `archerID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--
-- AUTO_INCREMENT for table `archer`
--
ALTER TABLE `archer`
  MODIFY `archerID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=49;

--
-- AUTO_INCREMENT for table `category_lookup`
--
ALTER TABLE `category_lookup`
  MODIFY `categoryID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=81;

--
-- AUTO_INCREMENT for table `class_lookup`
--
ALTER TABLE `class_lookup`
  MODIFY `classID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=17;

--
-- AUTO_INCREMENT for table `distance_lookup`
--
ALTER TABLE `distance_lookup`
  MODIFY `distanceID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;

--
-- AUTO_INCREMENT for table `division_lookup`
--
ALTER TABLE `division_lookup`
  MODIFY `divisionID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

--
-- AUTO_INCREMENT for table `end`
--
ALTER TABLE `end`
  MODIFY `arrowNo` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5521;

--
-- AUTO_INCREMENT for table `equivalence`
--
ALTER TABLE `equivalence`
  MODIFY `indexID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=23;

--
-- AUTO_INCREMENT for table `gender_lookup`
--
ALTER TABLE `gender_lookup`
  MODIFY `genderID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `range`
--
ALTER TABLE `range`
  MODIFY `rangeNo` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=461;

--
-- AUTO_INCREMENT for table `round`
```

```
--
ALTER TABLE `round`
  MODIFY `roundNo` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=116;

--
-- AUTO_INCREMENT for table `round_lookup`
--
ALTER TABLE `round_lookup`
  MODIFY `roundID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=14;

--
-- AUTO_INCREMENT for table `state`
--
ALTER TABLE `state`
  MODIFY `equivalenceID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT for table `targetface_lookup`
--
ALTER TABLE `targetface_lookup`
  MODIFY `targetFaceID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `archer`
--
ALTER TABLE `archer`
  ADD CONSTRAINT `archer_ibfk_1` FOREIGN KEY (`genderID`) REFERENCES `gender_lookup` (`genderID`);

--
-- Constraints for table `category_lookup`
--
ALTER TABLE `category_lookup`
  ADD CONSTRAINT `category_lookup_ibfk_1` FOREIGN KEY (`divisionID`) REFERENCES `division_lookup` (`divisionID`
  ADD CONSTRAINT `category_lookup_ibfk_2` FOREIGN KEY (`classID`) REFERENCES `class_lookup` (`classID`);

--
-- Constraints for table `club_championship`
--
ALTER TABLE `club_championship`
  ADD CONSTRAINT `club_championship_ibfk_1` FOREIGN KEY (`compID`) REFERENCES `competition` (`compID`);

--
-- Constraints for table `competition`
--
ALTER TABLE `competition`
  ADD CONSTRAINT `competition_ibfk_1` FOREIGN KEY (`roundNo`) REFERENCES `round` (`roundNo`);

--
-- Constraints for table `end`
--
ALTER TABLE `end`
  ADD CONSTRAINT `end_ibfk_1` FOREIGN KEY (`distanceID`) REFERENCES `distance_lookup` (`distanceID`),
  ADD CONSTRAINT `end_ibfk_2` FOREIGN KEY (`rangeNo`) REFERENCES `range` (`rangeNo`),
  ADD CONSTRAINT `end_ibfk_4` FOREIGN KEY (`targetFaceID`) REFERENCES `targetface_lookup` (`targetFaceID`) ON U
```

```

--
-- Constraints for table `equivalence`
--
ALTER TABLE `equivalence`
  ADD CONSTRAINT `equivalence_ibfk_1` FOREIGN KEY (`equivalenceID`) REFERENCES `state` (`equivalenceID`),
  ADD CONSTRAINT `equivalence_ibfk_2` FOREIGN KEY (`roundID`) REFERENCES `round_lookup` (`roundID`),
  ADD CONSTRAINT `equivalence_ibfk_3` FOREIGN KEY (`categoryID`) REFERENCES `category_lookup` (`categoryID`);

--
-- Constraints for table `range`
--
ALTER TABLE `range`
  ADD CONSTRAINT `range_ibfk_2` FOREIGN KEY (`roundNo`) REFERENCES `round` (`roundNo`);

--
-- Constraints for table `round`
--
ALTER TABLE `round`
  ADD CONSTRAINT `round_ibfk_1` FOREIGN KEY (`archerID`) REFERENCES `archer` (`archerID`),
  ADD CONSTRAINT `round_ibfk_2` FOREIGN KEY (`categoryID`) REFERENCES `category_lookup` (`categoryID`),
  ADD CONSTRAINT `round_ibfk_3` FOREIGN KEY (`roundID`) REFERENCES `round_lookup` (`roundID`);

--
-- Constraints for table `winner`
--
ALTER TABLE `winner`
  ADD CONSTRAINT `winner_ibfk_1` FOREIGN KEY (`archerID`) REFERENCES `archer` (`archerID`),
  ADD CONSTRAINT `winner_ibfk_2` FOREIGN KEY (`roundNo`) REFERENCES `round` (`roundNo`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

```

Document on data creation and null values (Week 7)

dataInsertion.sql
18 May 2023, 09:49 am

Overview

In the field of data management and database design, data creation and null values are two important concepts. Understanding them is essential to ensure the integrity, reliability, and accuracy of data stored in a database

Activity 1

Discuss in your group which attributes in your project database should be nullable.

We do not have nullable values in our database as we feel that it will make the database design complex and hard to manage.

Ambiguity: NULL can be ambiguous because it can represent several things – such as unknown, not applicable, or no value. It may not be clear to users or developers what NULL is supposed to represent in a specific context.

Complexity: Handling NULL values can add complexity to SQL queries. You need to use IS NULL or IS NOT NULL to check for NULLs, and functions like COALESCE to handle them. This can make queries harder to write and understand.

Indexing and performance: Most database systems exclude NULL values in indexes. If a column has a lot of NULL values and you're querying based on that column, it could impact the performance of your database queries.

Data integrity: Not allowing NULL values can help maintain data integrity. By enforcing columns to be NOT NULL, you are ensuring that every row entered into your database must contain a value for that column. This can prevent potential issues down the line where certain operations or calculations might fail due to unexpected NULL values.

Join operations: NULL values can complicate join operations. If you're joining on a column that contains NULLs, rows with NULL in that column won't be included in the result set.

Activity 2

Create the data for your database using your chosen tool, considering your decisions about nullable fields. 3000 entries in the smallest data table are sufficient (ignoring potential lookup tables, which may be smaller).

We decided to use https://www.mockaroo.com/ to generate most of the dummy data and added more than 3000 entries in our smallest data table (**end table**).

We managed to generate and add more than 5000 entries to our smallest data table, which is the 'end' table. This decision to exceed the minimum requirement of 3000 entries was intentional, as we aimed to stress-test our database under more substantial data loads. It's worth mentioning that the quantity of data in our lookup tables was less than 3000 entries due to their nature. The lookup tables only needed to contain a finite set of distinct values to serve as reference points for other tables in our database.

Activity 3

Decide on how best to upload the data and upload it into the team database.

We concluded that using SQL statements would be the most efficient and secure way to insert the data. This method allows us to directly interact with our database and review each operation's success or failure.

To ensure the integrity of our data and safeguard against potential errors during data insertion, we opted to use SQL transactions. By using transactions, we can perform multiple related operations as a single unit of work. If any part of the transaction fails, we have the option to roll back all the changes made during the transaction, thereby preserving the consistency and integrity of our data. This is particularly important when inserting large amounts of data, as it minimizes the risk of ending up with a partially updated or inconsistent database due to an error or failure during the insertion process.

Use cases and SQL statements, transactions (Week 8)

Adding archer:

```
INSERT INTO archer (firstName, lastName, dateOfBirth, genderID)
VALUES ('John', 'Doe', '1990-01-01', 1);
```

Update Archer's Info

```
UPDATE archer
SET firstName='Jane', lastName='Doe', dateOfBirth='1995-01-01', genderID=2
WHERE archerID=1;
```

Delete Archer

```
DELETE FROM archer
WHERE archerID=1;
```

Add New Round for an Archer

```
INSERT INTO round (archerID, date, roundNo, categoryID)
VALUES (1, '2023-06-01', 5, 3);
```

Update Round information of an Archer

```
UPDATE round
SET date='2023-07-01', roundNo=6, categoryID=4
WHERE roundID=1;
```

Delete Round

```
DELETE FROM round
WHERE roundID=1;
```

Add new Competition

```
INSERT INTO competition (compName, roundNo)
VALUES ('Spring Championship', 7);
```

List all archers in the database:

```
SELECT a.archerID, a.firstName, a.lastName, a.dateOfBirth, g.gender
                        FROM archer a
                        JOIN gender_lookup g ON a.genderID = g.genderID
```

Fetch round details using archer’s ID

```sql
SELECT r.roundID, r.date, r.roundNo, r.categoryID, cl.classDescription, dl.divisionDescription
               FROM round r
               JOIN category_lookup c ON r.categoryID = c.categoryID
               JOIN class_lookup cl ON c.classID = cl.classID
               JOIN division_lookup dl ON c.divisionID = dl.divisionID
               WHERE r.archerID = ?
```

Fetch competition, championship, class, division, and whether the archer has won or not

```sql
SELECT c.compName, cc.clubChampionshipName, cl.classDescription, d.divisionDescription, w.archerID AS winner
         FROM round r
         JOIN competition c ON r.roundNo = c.roundNo
         JOIN club_championship cc ON c.compID = cc.compID
         JOIN category_lookup cat ON r.categoryID = cat.categoryID
         JOIN class_lookup cl ON cat.classID = cl.classID
         JOIN division_lookup d ON cat.divisionID = d.divisionID
         LEFT JOIN winner w ON r.roundNo = w.roundNo AND r.archerID = w.archerID
         WHERE r.archerID = ?
```

Further use cases with logical design description

Personal best search:

Run a query for an archerID and a roundID order by highest to lowest on score and then only show top result

Multiple placing scoring for Competition

Run a query on the competition table to choose all records for a compID calculate the scores and order high to low

Multiple placing scoring for Club Championship

Run a query for a clubChampionshipID and then pull relevant compID, roundNo data to get scores, sum scores for an archerID, then order high to low to get placings

Score recorder can change a score from staging to official

An update query can be run against a roundNo to change `officalScore` from false (0) to true (1), queries looking to present rounds can check for records where officialScore = 1

There should be a definition of the default equipment, so that the category can be identified in the absence of user input

Run a query for the round_ID against the equivalence table, from there you can know which categories and therefore which divisions (equipment) are acceptable, the first option for the archerID's age calculation for class will be used to identify the default equipment

💪 Performance (indexes) (Week 9)

Indexes for the `archer` table

The primary key `archerID` is indexed to speed up any operation that involves searching, updating or deleting archers based on their ID. The `genderID` is also indexed because it's used frequently in JOIN operations, especially with the `gender_lookup` table.

Indexes for the `category_lookup` table

The `categoryID`, `divisionID` and `classID` fields are indexed. These fields are used for JOIN operations with other tables such as `round`, `class_lookup`, and `division_lookup` and the indexes will improve the performance of these operations.

Indexes for the `round` table

The `roundNo`, `roundID`, `archerID`, and `categoryID` are all indexed. The `roundNo` and `roundID` are part of the composite primary key and are frequently used in JOIN operations. The `archerID` and `categoryID` are foreign keys and are indexed because they are used frequently in queries and JOIN operations. `officialScore` is another field that has been indexed as this will be used frequently to sort staging and official scores.

Indexes for the `competition` table

The `roundNo` and `compID` are indexed because they are used in JOIN operations and in the SELECT statements to fetch competition and championship details.

Indexes for other tables

The primary keys of all other lookup tables (`class_lookup`, `division_lookup`, `gender_lookup`, `round_lookup`, `targetface_lookup`) are indexed because they are frequently used in JOIN operations and in the WHERE clause of SELECT statements.

Indexes for `range`, `end`, `equivalence`, `state`, and `winner` tables are created for similar reasons. The fields that are part of JOIN operations, part of the WHERE clause in SELECT statements, or are used in ORDER BY, GROUP BY clauses are indexed to speed up these operations.

In our case, the decision to create these indexes is based on the frequent use of these fields in our use cases (the SQL statements provided). These indexes should provide a significant performance improvement for these operations.

SQL Code:

```
--
-- Indexes for table `archer`
--
ALTER TABLE `archer`
  ADD PRIMARY KEY (`archerID`),
  ADD KEY `genderID` (`genderID`);

--
-- Indexes for table `category_lookup`
--
ALTER TABLE `category_lookup`
  ADD PRIMARY KEY (`categoryID`),
  ADD KEY `divisionID` (`divisionID`),
  ADD KEY `classID` (`classID`);

--
-- Indexes for table `class_lookup`
--
ALTER TABLE `class_lookup`
  ADD PRIMARY KEY (`classID`);

--
-- Indexes for table `club_championship`
--
ALTER TABLE `club_championship`
  ADD PRIMARY KEY (`compID`);

--
-- Indexes for table `competition`
--
ALTER TABLE `competition`
  ADD PRIMARY KEY (`roundNo`),
  ADD KEY `compID` (`compID`);

--
-- Indexes for table `distance_lookup`
--
ALTER TABLE `distance_lookup`
  ADD PRIMARY KEY (`distanceID`);

--
-- Indexes for table `division_lookup`
--
ALTER TABLE `division_lookup`
  ADD PRIMARY KEY (`divisionID`);

--
-- Indexes for table `end`
--
ALTER TABLE `end`
  ADD PRIMARY KEY (`arrowNo`) USING BTREE,
  ADD KEY `distanceID` (`distanceID`),
  ADD KEY `targetFaceID` (`targetFaceID`),
  ADD KEY `rangeNo` (`rangeNo`);

--
-- Indexes for table `equivalence`
--
ALTER TABLE `equivalence`
  ADD PRIMARY KEY (`indexID`),
  ADD KEY `equivalenceID` (`equivalenceID`),
  ADD KEY `roundID` (`roundID`),
  ADD KEY `categoryID` (`categoryID`);

--
-- Indexes for table `gender_lookup`
--
ALTER TABLE `gender_lookup`
  ADD PRIMARY KEY (`genderID`);

--
```

```sql
-- Indexes for table `range`
--
ALTER TABLE `range`
  ADD PRIMARY KEY (`rangeNo`),
  ADD KEY `roundNo` (`roundNo`);

--
-- Indexes for table `round`
--
ALTER TABLE `round`
  ADD PRIMARY KEY (`roundNo`,`roundID`),
  ADD KEY `archerID` (`archerID`),
  ADD KEY `categoryID` (`categoryID`),
  ADD KEY `roundID` (`roundID`);

--
-- Indexes for table `round_lookup`
--
ALTER TABLE `round_lookup`
  ADD PRIMARY KEY (`roundID`);

--
-- Indexes for table `state`
--
ALTER TABLE `state`
  ADD PRIMARY KEY (`equivalenceID`);

--
-- Indexes for table `targetface_lookup`
--
ALTER TABLE `targetface_lookup`
  ADD PRIMARY KEY (`targetFaceID`);

--
-- Indexes for table `winner`
--
ALTER TABLE `winner`
  ADD PRIMARY KEY (`compID`),
  ADD KEY `archerID` (`archerID`),
  ADD KEY `roundNo` (`roundNo`);

```

💪 Team reflection (Week 12)

📋 Overview

Reflect back on what you and your team learned and what motivates the group to succeed by following the instructions for the 4Ls Retrospective Play.

Team	Alpha Team
Team members	@Yap Zhe Wei @Ben Wierenga @Nguyen Nam Tung @Lohan Thilakarathna @Tashahud Ahmed
Date	11 May 2023
Retrospective period	27 Feb 2023 - 1 Jun 2023

💭 4Ls retrospective

Milestones	Loved	Longed for	Loathed	Learned
ARCHERY DATABASE	• When SQL statements work as expected • Resolving FK constraints issue in database	• More resources on database design	• When SQL statements doesn't work • Normalisation of database	• Design thinking • Database Normalisation • Database Design
PRESENTATION VIDEO	• Visually appealing slides • Sufficient information being presented within time limit	• Interactive Powerpoint slides	• Plans getting delayed	• Effective communication with teammates
MAJOR RELATED WORK	• When my website displays all the results as expected • Database visualisation is accurate and interactive • Get to demonstrate skills that we have learned	• Demonstrating my skills • More time to explore more data visualisation • Implementing more features and functionalities to the website	• Configuration issues with database (Data Visualisation) • Repetitive work with implementing different data visualisation • When the arrangement of the website components is out of place	• Efficient collaboration with teammates • Analysing database and presenting it • Displaying results on the website from database • Create a user friendly and interactive interface

⚡ Action plan

Action	Owner	Due date	Action items
Improve my understanding of database configuration and data visualization tools. I'll dedicate some time each week to learn more about common configuration issues and ways to troubleshoot them.	@Nguyen Nam Tung @Lohan Thilakarathna	1 Jun 2023	Configuration issues with database (Data Visualisation)
Invest more time in learning advanced SQL techniques and concepts. I might use online resources or enroll in a relevant course. This could improve my efficiency and effectiveness in writing SQL queries, and possibly result in having more of those successful moments.	@Yap Zhe Wei	1 Jun 2023	When SQL statements work as expected
Seek out and utilize more resources to expand my knowledge. This could include books, online courses, and webinars from experts in the field. A broader understanding would not only help me in my current role but also in the long term as I continue to grow in my career.	@Ben Wierenga	1 Jun 2023	More resources on database design
To amplify this, I will initiate regular team-building activities or knowledge-sharing sessions within the team. This will not only enhance our collaboration but also give us a platform to share valuable insights and learnings.	@Lohan Thilakarathna @Tashahud Ahmed	1 Jun 2023	Efficient collaboration with teammates

👥 Contributions

Major-specific work:

🖱️ Software Development Major-related Work – @Yap Zhe Wei & @Tashahud Ahmed

💻 Cybersecurity Major-related Work - @Ben Wierenga

📕 Data Science and Analysis Major-related work – @Nguyen Nam Tung & @Lohan Thilakarathna

@Yap Zhe Wei

Item	Contribution
Confluence	Writing Confluence pages (Weekly Meeting minutes, etc) Adding Confluence pages (Team health monitor, progress documentation, etc.) Add documentation Update Confluence weekly
Jira	Create Jira template Add Jira Tasks Update Jira weekly
Database design	Contributed to creation and ideation of the database Add dummy data
Website Design	Made the website to display data Create SQL statements to display data Connect website to database Collaborated with @Ben Wierenga for sanitisation Added functionalities to website (date filter, data retrieval, etc.) Styled the website 🖱️ Software Development Major-related Work
Presentation Video	Writing and arranging slides Record voiceover for slide

@Ben Wierenga

Item	Contribution
Confluence	Adding Confluence pages Performing initial setup
Jira	Arranging tasks for epics Performing initial setup
Database design	Organising the hosting and access for the project Working heavily on database table layout Assigning permissions for both users and the website functionality from a Cybersecurity perspective
Website Design	Organising the website hosting and functionality Arranging for input sanitisation Collaboration with @Yap Zhe Wei on website structure and creating prepared statements (IN PROGRESS)
Presentation Video	Writing many of the slides Stitching clips together for the submission

@Nguyen Nam Tung

Item	Contribution

Confluence	Writing Confluence Page Edit Confluence Page
Jira	Complete Jira Task
Database design	Creating the ER diagram and update them gradually Initial idea for the database Database Layout Creation Inserting Dummy Data Fixing the key of the database (Foreign key, primary key)
Archery Database Analysis	Adjust the relationships in the database for the tables to create Create all the Archery Database Dashboard GIve details comments and analysis for the dashbord The detail dashbord: Data Science and Analysis Major-related work
Presentation Video	Add information to the slide

@Lohan Thilakarathna

Item	Contribution
Confluence	Added and edited pages
Jira	Added information to Jira tasks
Database design	Helped to figure out look up tables and SQL needed for functionality. Added dummy data
Database analysis	Helped set up and configure Power BI. Helped @Nguyen Nam Tung with dashboard
Presentation Video	Added information to slides

@Tashahud Ahmed

Item	Contribution
Confluence	Added pages Edited pages
Jira	Added information Update tasks
Database design	Inserted dummy data Assisting with fixing the keys of the database (primary and secondary datas)
Website Design	Edited styles Added to styles
Presentation Video	Writing slides

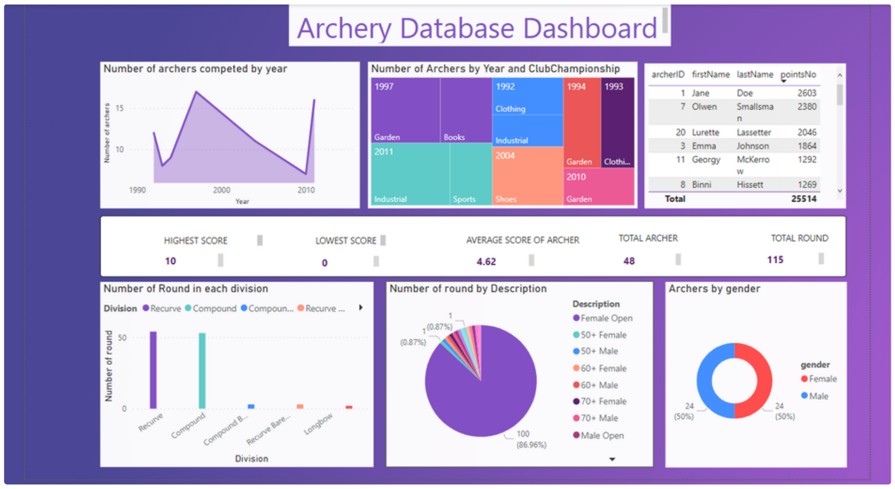
Data Science and Analysis Major-related work

Contributions – @Nguyen Nam Tung & @Lohan Thilakarathna

Here is a dashboard showing some analysis of the archer database. It can be download here: (in pdf if you just want to see the overall structure or in power bi file (pbit) if you want to see the interactive feature and play around with the data)

ArcherDashboard.pbit
15 May 2023, 03:37 am

ArcherDashboard.pdf
15 May 2023, 03:17 am



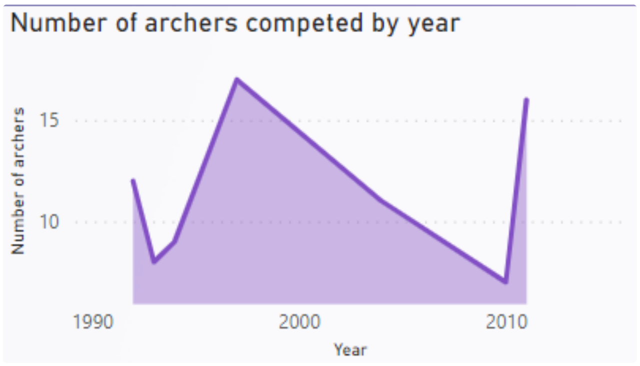
archerID	firstName	lastName	pointsNo
1	Jane	Doe	2603
7	Olwen	Smallsman	2380
20	Lurette	Lassetter	2046
3	Emma	Johnson	1864
11	Georgy	McKerrow	1292
8	Binni	Hissett	1269
Total			25514

HIGHEST SCORE	LOWEST SCORE	AVERAGE SCORE OF ARCHER	TOTAL ARCHER	TOTAL ROUND
10	0	4.62	48	115

Archery Database Dashboard

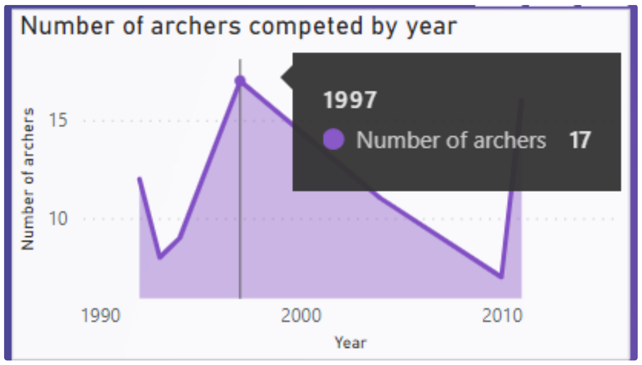
The visualization dashboard was created using Power BI. Here is the full description of the dashboarh and some analysis from it

Number of archers competed by year



Area chart showing archers by year

This is an area chart that demonstrates the number of archer competed by year. There are a lot of fluctuation for the figure, and the highest one is in year 1997, with 17 archer (total archer is 48)



Highest figure for archer competed

After reaching the peak, it witnessed an extreme decline to 7 in year 2010, and then greatly increased to 16 archer in year 2011

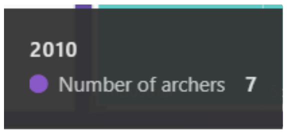


Figure for 2010

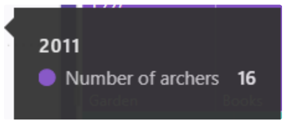
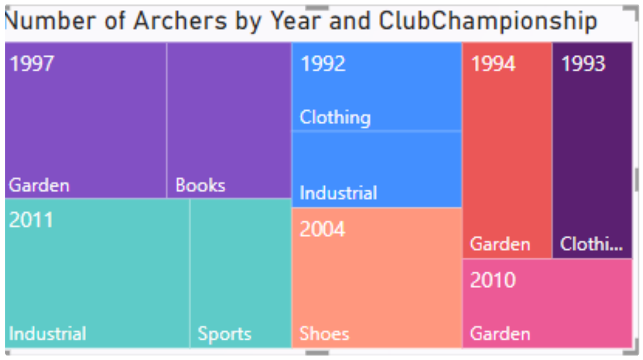


Figure for 2011

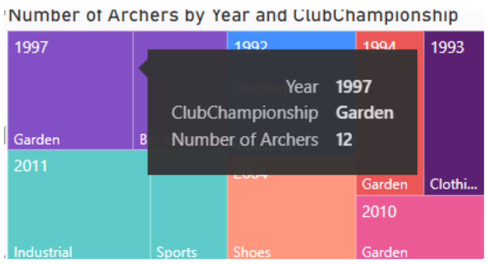
Number of archers by year and club championship



Treemap showing Number of Archers by Year and Clubchampionship

The next visualization is a treemap showing the number of archer by year and club championship. The name of club championship is the name of very basic things in our life: Garden, Book, Clothing, …

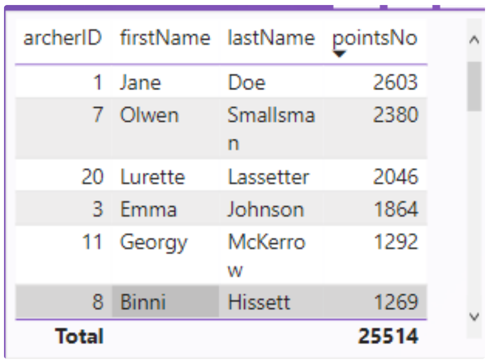
In this diagram, you can not see clearly the figure, but in the pbit file, you can see the interactive feature when you hover your mouseover. For instance: When you hover on 1997-Garden, this will show up:



Interactive feaure of tree map

Table listing the archers and their total score, from the highest to lowest

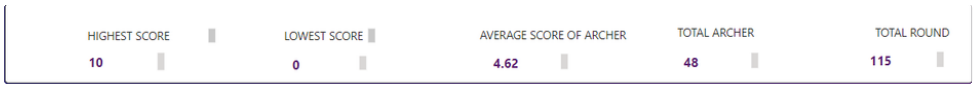
It is basicly showing the top archer with the highest score



archerID	firstName	lastName	pointsNo
1	Jane	Doe	2603
7	Olwen	Smallsman	2380
20	Lurette	Lassetter	2046
3	Emma	Johnson	1864
11	Georgy	McKerrow	1292
8	Binni	Hissett	1269
Total			**25514**

Table listing archers and their score

Some interesting statistics



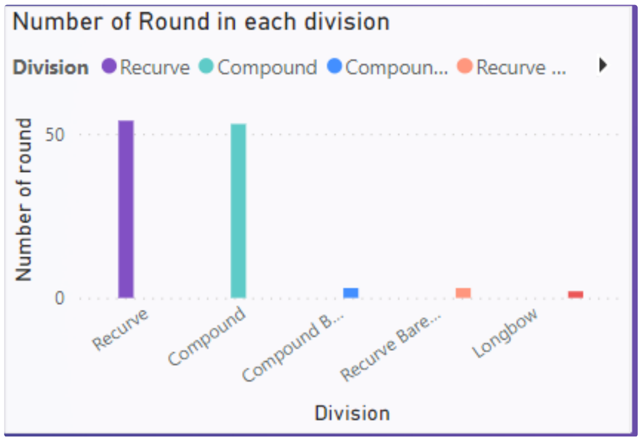
HIGHEST SCORE	LOWEST SCORE	AVERAGE SCORE OF ARCHER	TOTAL ARCHER	TOTAL ROUND
10	0	4.62	48	115

Interesting statistics

This part shows some statistic such as highest, lowest score for each end, average score of archer, …

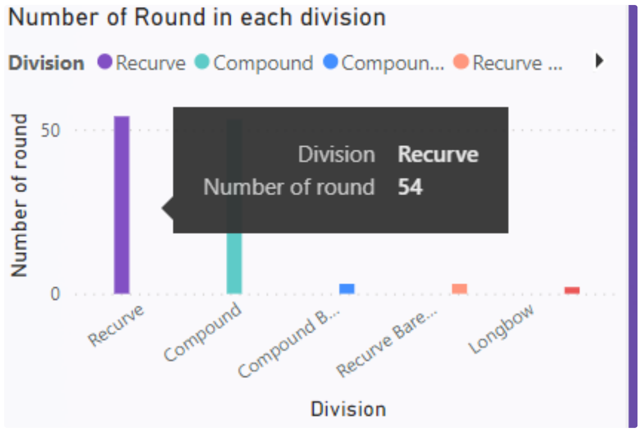
Number of round in each division

This part is a bar chart that show the number of round in each division



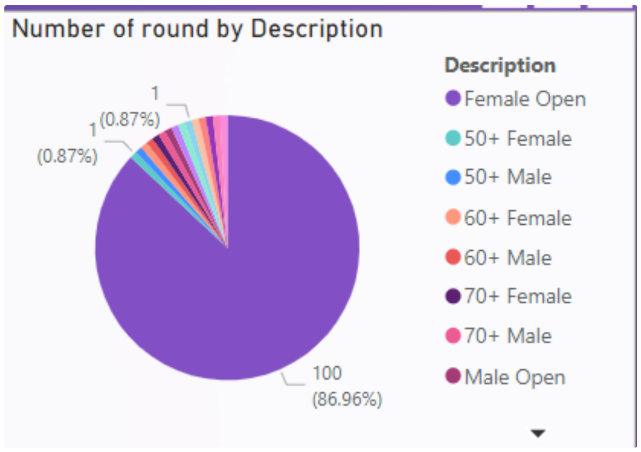
Bar chart showing the round in each division

Recurve bow has the highest number of round with 54 rounds



Highest figure

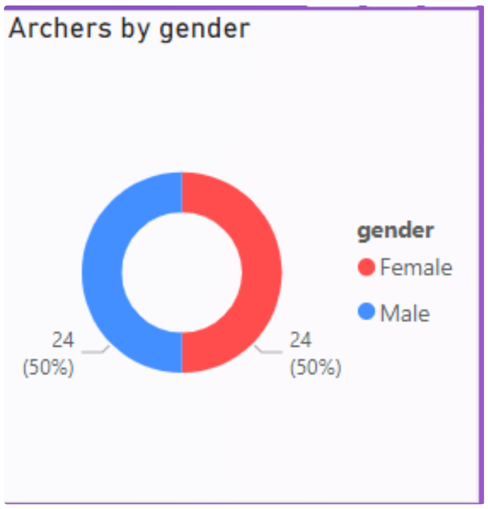
Number of round by Description



Pie chart show the number of round by description

The pie chart shows the number of round in each description. As you can see, Female Open has the highest number of rounds (100 with 86,69% of all round)

Archer by gender



Donut chart shows archers by gender

This is a donut chart that show archers by gender. As you can see from the chart, the number of female and male players are the same (24)

💻 Cybersecurity Major-related Work

Contribution – @Ben Wierenga

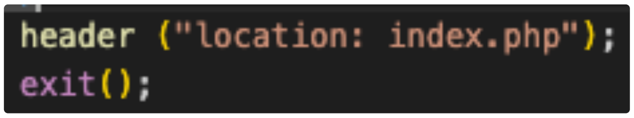
Within the context of Cybersecurity we modified our database to provide access controls, and added input sanitisation to prevent SQL injection. Collectively these work to prevent some standard intrusion attempts for a typical website. Acting to stop these lower level functions ensures that malicious actors cannot compromise the server and then escalate privileges to allow for lateral movement across the Archery club’s network.

The primary goal is to ensure the integrity of the database, meaning that any recorded scores are safe from tampering and the competitions which are viewed across Australia or the world are reliable and can be used for comparing to personal bests, club, national, and world records.

Implementation of these protections has been done on both the construction of the php pages used to create the interactive web front end from the Software Development major specific work. 👥 Contributions

Protection in our PHP code

We created a test page to allow us to confirm that our queries were working but also needed to keep it available for our use, so we added a redirection that can easily be commented out for when we need to test functionality.



```
header ("location: index.php");
exit();
```

Code to allow easy redirection from the test page address at http://3.131.96.127/test.php

Additional measures were taken to in the PHP code, primarily to prevent SQL injection but these measure also accomplish a secondary goal of reducing errors that might present from special characters being entered into input fields.

```
//Sanitise inputs function
                function sanitise_input($data) {
                    $data = trim($data);
                    $data = stripcslashes($data);
                    $data = htmlspecialchars($data);
                    return $data;
                }
```

This special function will take any inputted user data and remove trailing spaces, remove backslashes and convert any remaining special characters to HTML entities that cannot affect the code that processes the entity. Resulting in SQL injections failing when attempted against the input fields that are substituted into the pre-constructed SQL queries, leading to our second cybersecurity innovation for our project.

Use of Prepared Queries

Whilst using html sanitisation for user input helps prevent some aspects of SQL injections, using binded parameters for submitted database queries completes the protections we have implemented.

```
//Bind parameters to the SQL statement
                    if (!empty($placeholders)) {
                        $paramTypes = str_repeat("s", count($placeholders));
                        $bindParams = array($stmt, $paramTypes);
```

```
                              foreach ($placeholders as &$param) {
                                  $bindParams[] = &$param;
                              }
                              call_user_func_array('mysqli_stmt_bind_param', $bindParams);
                         }
```

The above snippet allows the search criteria added by the user input to be filled in as blanks to a prepared statement. This has two clear benefits.

First, every time a query is run on the database, the database management system (DBMS) needs to parse the query and then resolve what data to return. This process is completed for every new query structure the DBMS has to process. Therefore, by reusing the same query structure with different variables the efficiency of the database can be increased.

Secondly, this creates clear blanks in the prepared statement that are filled with user input but prevent unexpected information from overflowing into the rest of the instruction, eliminating the potential for SQL injection entirely.

Thus the combination of protections like input sanitisation and prepared statement we are able to secure our web front end from SQL injection, leading to the protection of the confidentiality of user data and the integrity of the database. Meeting the two relevant pillars of cybersecurity. Access (the remaining pillar) can be met with a redundant front end using load balancers and multiple IPs however, this remains out of scope for the implementation and project requirements for an Archery database.

Permission Constraints on Tables

Security is built on the “Principle of Least Privilege” where users are not given more access than they need. This prevents two key things, one; users cannot see and therefore know about things they do not need to interact with, preventing the unintentional release of information. Two; when malicious actors are hunting for information they know or assume to exist they will be prevented from finding it assuming least privilege has been implemented properly.

In the case of this project a “data_entry” user has been created for the purposes of the back end data interaction between the web server and the database server.

To implement least privilege in our scenario, this user has been granted the following permissions

```
GRANT SELECT ON `COS20031`.* TO `data_entry`@`localhost`;
```

This allows them to read all tables needed for calculations. However if we implement a login page in a future update, this user would not be given SELECT for that table and any related tables. Instead a separate login would be used to allow for separation of powers (another principle).

To allow for the uploading of scores to the database, additional granular permissions have been granted as per below.

```
GRANT INSERT ON `COS20031`.`round` TO `data_entry`@`localhost`;

GRANT INSERT ON `COS20031`.`end` TO `data_entry`@`localhost`;
```

With only limited insert commands to the minimum number of tables, the principle of least privilege has been applied correctly. An additional update permission can be granted to the round table when and only when the functionality is added to the front end web server to allow for an update to existing entries to ‘confirm' scores as official.

🖱️ Software Development Major-related Work

Contributions – @Yap Zhe Wei & @Tashahud Ahmed

We created a web-based system that includes two primary pages: the initial landing page (index.php) and the detailed archer profile page (details.php). The system also uses a style.css file for a pleasing aesthetic design.

Here is the link to the website: http://3.131.96.127/index.php

Files

index.php

The main landing page, index.php, greets visitors with a search feature that enables them to find archers using various criteria such as first name, last name, archer identifier, or gender. When the search function is activated, it pulls relevant entries from the database and presents them in a structured table format. Each table row contains basic details about an archer and a clickable link for in-depth information.

To enhance user experience, this page integrates a pagination feature that limits the number of results displayed on a single page. This, along with the search feature, makes use of the GET method, enabling visible URL display of search parameters and current page numbers.

details.php

The details.php page offers a comprehensive profile view for a specific archer. It is accessed via the 'View Details' link from the index.php page. The details include a wide range of information about the archer, such as personal data, recorded rounds, and their history in various competitions and championships.

If no valid archer identifier is provided, the system displays an appropriate error message. Furthermore, a 'Back to Archers' link is present at the end of the page to facilitate easy navigation back to the index.php page.

enter_score.php

This page consist of a comprehensive user friendly interface that allows users to record their scores. User will be prompted to enter their details such as, Round, Division, Archer(Their name), Ends, Distance(m), and Target Face (cm).

After entering their details, they can also enter their score in the “Enter your score” section, the scores they entered will be calculated automatically after clicking the submit button, the results will be recorded in the table below this section.

style.css

The style.css file provides a straightforward CSS template that provides styling to elements like the body, h1, and h2 for a sleek and professional presentation.

Most of our web pages are styled using the Bootstrap CDN that we have implemented.

Database

The web system interacts with an archery-specific database that houses tables for various archery elements, including archers, genders, rounds, categories, classes, divisions, competitions, championships, and winners. Prepared statements are used for querying the database as a measure against potential SQL injection threats. The PHP mysqli extension is used for interfacing with the MySQL database.

Querying

On the index.php page, user-input search parameters dictate how the database is queried. If no specific parameters are given, a list of all archers is displayed. On the other hand, the details.php page queries the database for detailed information on a single archer, pulling up data about the rounds they've participated in and their performance in competitions and championships.

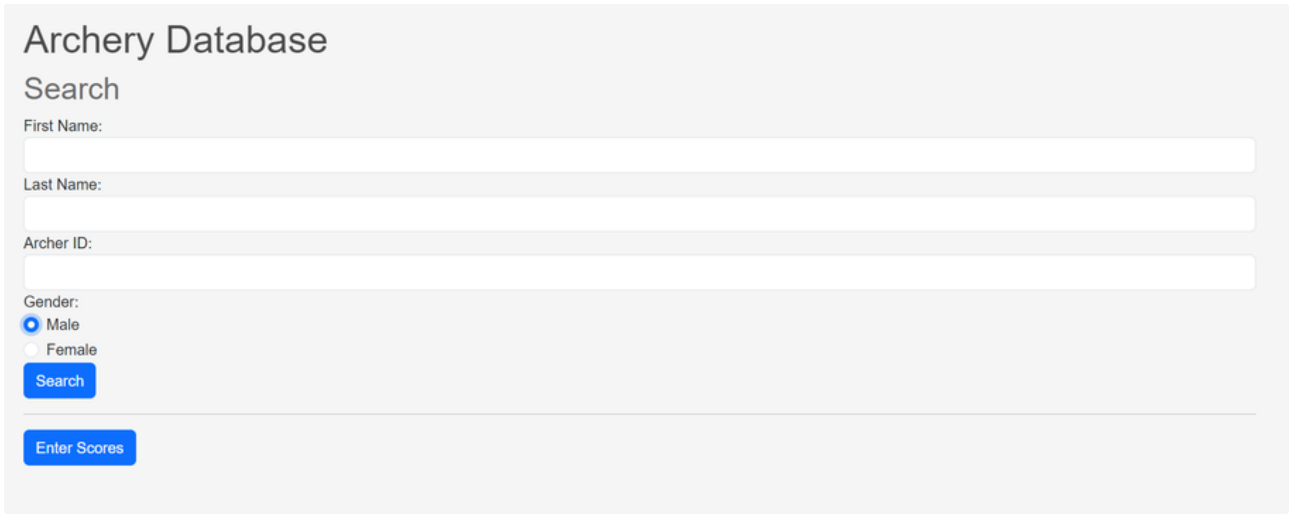
We also used SQL queries to list out all the necessary details in the enter_score.php to populate the dropdown menu, this eliminates the need to update the data in the dropdown menu manually as it will be dynamically populated using the database.

Error Handling

A rudimentary error-handling system is in place. In the event of a failed database connection, an error message is shown. In cases where a SQL statement fails to prepare or execute, the error is displayed. When an archer can't be found, appropriate messages such as 'No archers found' or 'Archer not found' are displayed. If the details.php page is accessed without a proper archer identifier, a 'No Archer ID provided' message is shown.

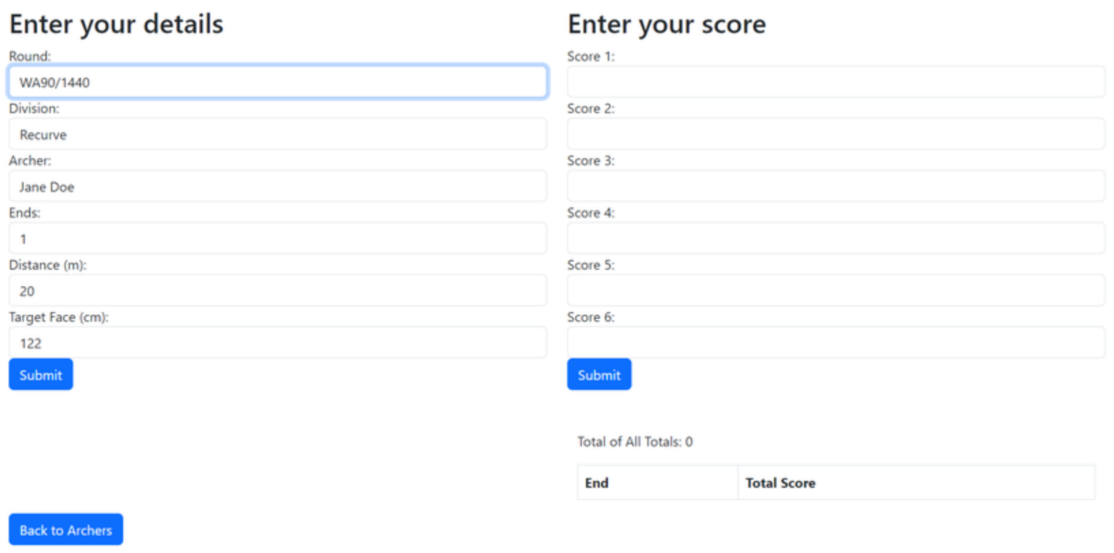
Website Functionalities and Usage

This is how the website starting page looks like:



Website starting page

The website allows the users to search for archers ' records based on First Name, Last Name. Archer ID, Gender. The archer can also enter their scores to the website. When the users want to enter their scores, a data inputting page will show up that allows them to do so.



Enter your details

Round:	WA90/1440
Division:	Recurve
Archer:	Jane Doe
Ends:	1
Distance (m):	20
Target Face (cm):	122

Submit

Enter your score

Score 1:

Score 2:

Score 3:

Score 4:

Score 5:

Score 6:

Submit

Total of All Totals: 0

End	Total Score

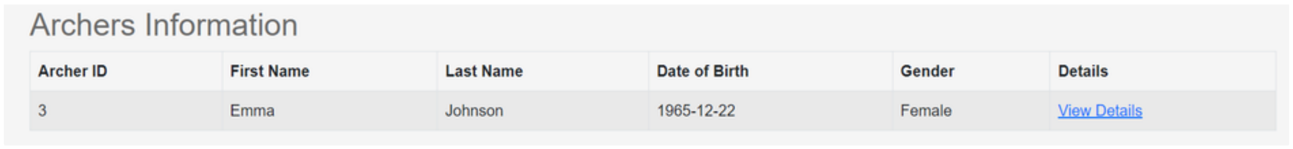
Back to Archers

Enter Score page

This page allows the archers to put in their scores for their correcsponding Round, Division, Ends, Distance, Target Face.

The archers can view their records. For instance a player \with id 3 want to see his record.

A record will show up

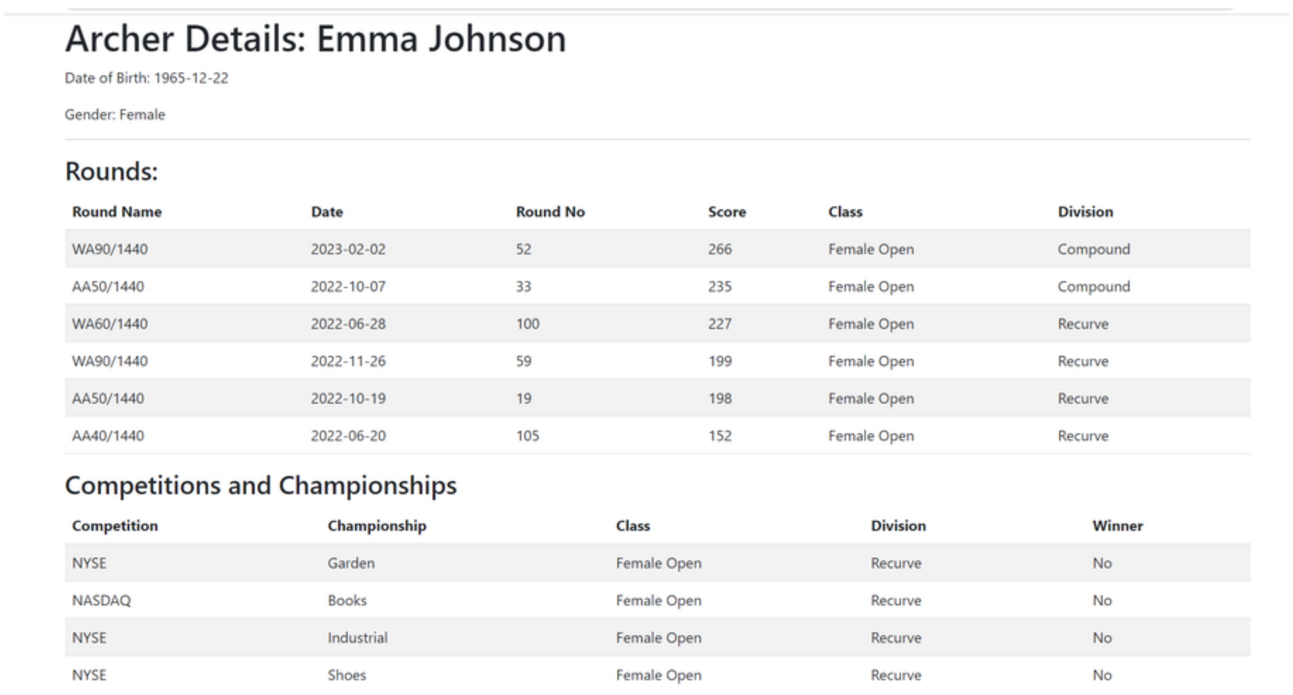


Archers Information

Archer ID	First Name	Last Name	Date of Birth	Gender	Details
3	Emma	Johnson	1965-12-22	Female	View Details

Archer information

The archer can see his details and his detailed archery record like Date, Score, Class



Archer Details: Emma Johnson

Date of Birth: 1965-12-22

Gender: Female

Rounds:

Round Name	Date	Round No	Score	Class	Division
WA90/1440	2023-02-02	52	266	Female Open	Compound
AA50/1440	2022-10-07	33	235	Female Open	Compound
WA60/1440	2022-06-28	100	227	Female Open	Recurve
WA90/1440	2022-11-26	59	199	Female Open	Recurve
AA50/1440	2022-10-19	19	198	Female Open	Recurve
AA40/1440	2022-06-20	105	152	Female Open	Recurve

Competitions and Championships

Competition	Championship	Class	Division	Winner
NYSE	Garden	Female Open	Recurve	No
NASDAQ	Books	Female Open	Recurve	No
NYSE	Industrial	Female Open	Recurve	No
NYSE	Shoes	Female Open	Recurve	No