# Programming Assignment 2 Report

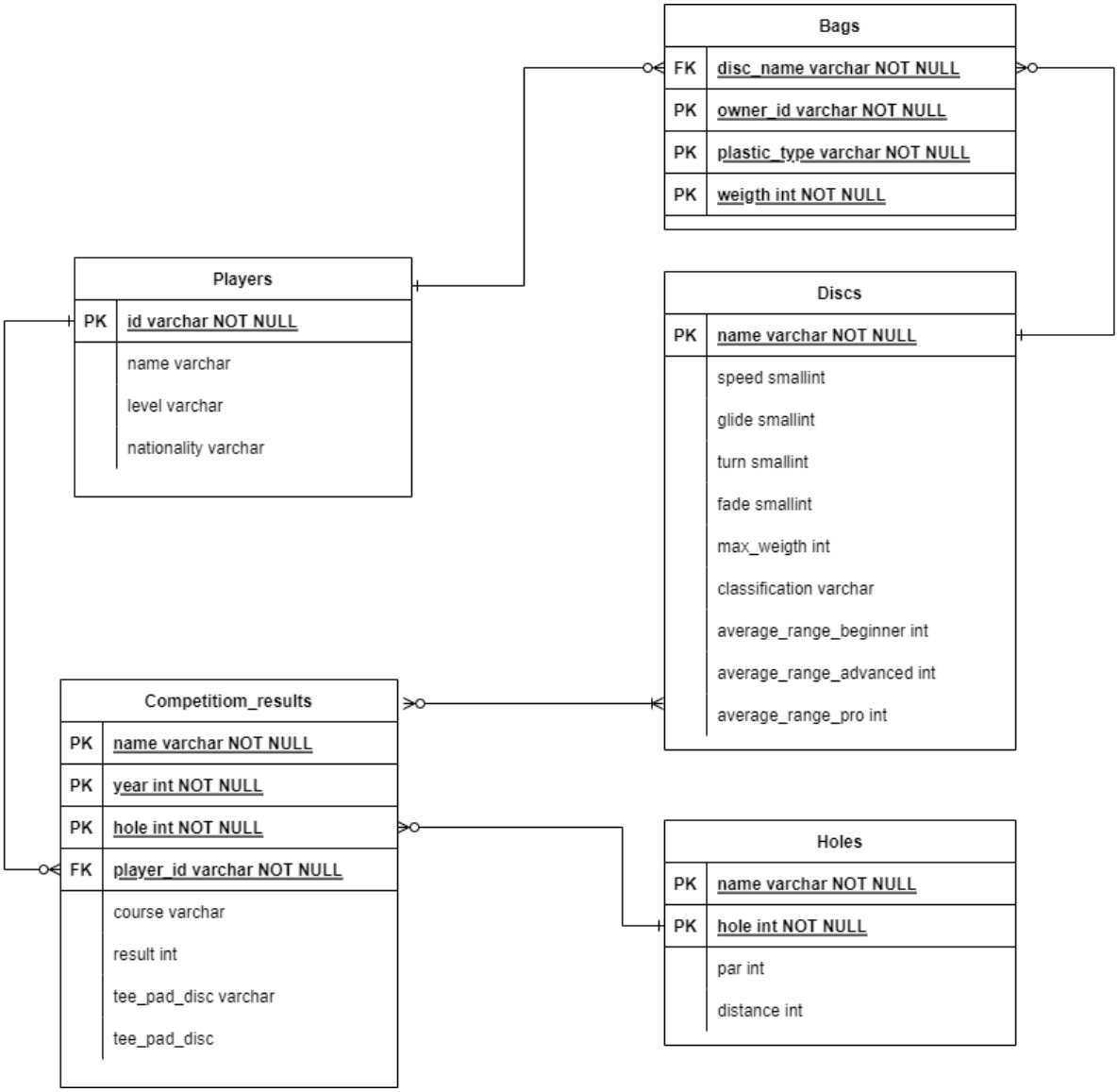

| | |
|---|---|
| Student(s): | Johan Gustafsson – jg223fp@student.lnu.se<br>Anas Hallak Mohamadanas hm222ua@student.lnu.se |


## 1. Project Idea

We have developed a database for disc-golfers. It contains data about different discs, courses, players, and results. Some of the data is made up, for example competitions, players, and results. We have collected some data about different discs from a company named Latitude64 which produce a variety of different discs. The data about the courses is taken from a homepage called Udisc.
Our app lets you see what discs different players have in their possession. It can also tell you which discs would be good if you want to throw further than a specific distance depending on you level. Of course, it can also calculate the winner of competitions. Another nice feature is that it can tell which is the by players most owned disc. This could be of great in market research.

Data sources:
Latitude64: https://www.latitude64.se/disc-golf-products/golf-discs/
Teleborgs disc golf course: https://udisc.com/courses/teleborgs-discgolfbana-i7Em
Älmhults disc golf course: https://udisc.com/courses/aelmhults-discgolfbana-eNDA

# 2. Schema Design




*Picture 1: Schema design for the disc golf database*

# Tables explanations

## Players

This table represents a disc golf player. The have a player id, a name, and a nationality. They are also ranked in different levels. The id is the primary key to identify a player.

## Discs

This table contains facts about different disc models. They have different flying attributes which describes how they behave in the air. Those are speed, glide, turn, and fade. The discs are also classified for their intended usage in driver, fairway driver, midrange and putts. The table also provides data of the average throwing distance for each disc for each different player level. The primary key is the name of the disc model

## Holes

A record in the holes table is a hole on a course. A course usually has 9 or 18 holes. Each hole has a distance and also a par result. A par is the result you are aiming to beat when playing the hole. To identify a specific hole the primary keys are the name of the course and the hole number.

## Bags

The table “Bags” are actually discs that is owned by someone and are put in bags. The name can be a bit confusing but the table consist of discs with different owner id’s and some specific attributes which identifies the disc. Those attribute is plastic type and weigth. Discs are sold in different materials and can have different weigth depending on if you are strong or weak. All the attributes is primary key except the disc name which is a foregin key coming from the name attribute in the discs table.

## Competition_results

The records in competition_results are actually the result for each player for each hole in a competition. In each record we can also se what discs where used at throw from tee pad and the disc which was used when finishing the hole. The primary keys are name of competition, year, and hole number. The player id is a forigen key and is referenced to the id in the player table. the The course is not needed for primary key as a competition can be played on different courses.

## Relations explanations

- A bagged disc has one owner id and one player can own zero or many discs.
- A competition result has one player and a player can have zero or many competition result.
- A competition result have one hole. A hole can have zero or many competition results.
- A competition result has one or many discs (disc used on tee pad and disc used to finish the hole) and a disc can be used in zero or many results.
- A disc in a bag is of one specific disc model and a disc model can exist in zero or many players bags.

## 3. SQL Queries

Q: **Who is the winner of a given competition?**
The following query is multirelational and uses JOIN. The program will first send a query to list all competitions. Then it lets the user select a competition and pass the name (marked with ? in the query) and year (marked with X in the query)  to it on the the following query. The query sums the result of the given competition and picks the one with the lowest result. Then it joins the result on the foreign key *player_id* with *id* the player table to get facts about the winner and return together with the result.

```
SELECT name,id,nationality,total FROM disc_golf.players
JOIN
(SELECT
    player_id, SUM(result) AS total
FROM
    disc_golf.competition_results
WHERE
    year = X
        AND competition_results.name = ?
GROUP BY player_id
ORDER BY total
LIMIT 1) as winner
ON players.id = winner.player_id;
```

Q: **Which discs of a given class can be thrown longer than a given distance by a given level of player?**
There are discs that are good for beginners, some for advanced players, and some for pros. All discs have a specific average range for all three levels. This query reads a classification of the disc, a player level, and also a given distance. It returns the discs which can be thrown further than the given distance and which has the right classification.

```
SELECT name FROM disc_g_golf.discs
WHERE classification = provided classification
AND provided_player_level >= provided_distance;
```

Q: **Which disc classification is the most used one on tee pads for a given course?**
Different courses can make use of different disc classes. E.g., a course with short holes can be played only using putts and for a course with long holes you need to use a driver. This query lets the user select a disc-golf course and then tells the most used disc classification on tee pads based on competition results. This query uses grouping and aggregation. It is also a multi-relational query and begins by counting the number of times a certain disc is used on the tee pads on the provided course. It then returns the name of the most used disc and then it compares it with the name from the discs table. Then it can select the classification of the disc.
The course passed on is marked with a `?` in the query.

```
SELECT
    classification
FROM
    disc_golf.discs
WHERE
    name = (SELECT
            tee_pad_disc
        FROM
            (SELECT
                tee_pad_disc, COUNT(tee_pad_disc) AS discCount
            FROM
                disc_golf.competition_results WHERE
competition_results.course = ?
            GROUP BY tee_pad_disc
            ORDER BY discCount DESC     # sort in falling order
  LIMIT 1) AS mostUsed);                # get the first result
```

Q: **Which disc is owned by the most players?**
This query is multi relational and uses aggregation and grouping. It counts the most frequent disc name in the player bags. It then returns the most frequent name and selects the attributes of that disc from the *disc* table.

```
SELECT
    name,
    classification,
    speed,
    glide,
    turn,
    fade,
    average_range_beginner,
    average_range_advanced,
    average_range_pro
FROM
    disc_golf.discs
WHERE
    name = (SELECT
            disc_name
        FROM
            (SELECT
                disc_name, COUNT(disc_name) AS discCount
            FROM
                disc_golf.bags
            GROUP BY disc_name
            ORDER BY discCount DESC
            LIMIT 1) AS mostOwned);
```

Q: **What discs does a given player own?**
This query lets you se what discs different players own and keep in their bag. This can be interesting to see if a player is good and inspiring.
First the players are listed by a query and the user can select a player. Another query then fetches the *id* of the selected player (marked with ? in the query).
The query joins the bag records on the *disc_name* with *name* from the discs table and also where the given *id* appears.

```
SELECT disc_name,
       plastic_type,
       weigth,
      speed,
      glide,
      turn,
      fade,
      classification
FROM
            disc_golf.bags
JOIN disc_golf.discs
ON discs.name = bags.disc_name
    AND bags.owner_id = ?
ORDER BY classification;
```

Q: **Which hole is the longest in a given competition?**
In this query the user can select a competition from a list of competitions brought by another query. The query then joins on *competition_result.course* with *name* from the *holes* table. It then orders the holes by distance in descending order and return the one at the top. The competition name is marked with ? in the query and year is marked with X . It is multi relational and also uses join.

```
SELECT competition_results.name,
      year,
      course,
      holes.hole,
      par,
      distance
FROM disc_golf.competition_results
JOIN holes ON competition_results.course = holes.name
 AND competition_results.hole = holes.hole
 AND competition_results.name = ?
 AND year = X
ORDER BY distance DESC
limit 1;
```

Q: **Can I have all the information about the players?**
This query uses the view “playerinfo” to fetch all the info about the players. It currently doesn’t have any restrictions. It is used as part of looking inside players bags when listing all the players.

```
SELECT * from playerinfo;
```

## 4. Discussion and Resources

The tough part for this project was to get data. In the end we ended up creating all the data manually and entering it in to excel documents. Some data was as earlier mentioned taken from websites but some of it was just fabrication. It is also the first project we build which uses a GUI. This has been both fun and time consuming. It sure adds a feeling of a more complete app in the end.

The projects use the following resources that needs to be installed:

- Python version >= 3.10
- Pandas
- PySimpleGUI
- Mysql.connector

Please check readme.md for installation details.

Source code: https://github.com/jg223fp/database_assignment3
Video demonstration: https://youtu.be/TVl4wMAwu5A

# Changelog

| Person | Task | Date |
|---|---|---|
| **Johan** | Setting-up server environment and Git repository | 2022-03-04 |
| **Johan** | Created the e/r diagram and schema | 2022-03-04 |
| **Johan** | Added data to CSV files | 2022-03-06 |
| **Johan** | Wrote parser.py which parses data into the server | 2022-03-07 |
| **Johan** | Tested and prepared all queries in MySQL workbench | 2022-03-08 |
| **Johan** | Divided project into view, controller and parser | 2022-03-08 |
| **Johan** | Finished all queries and wrote methods for them in controller.py | 2022-03-09 |
| **Johan** | Began writing report and edited schema | 2022-03-09 |
| **Anas** | Began creating GUI | 2022-03-09 |
| **Johan** | Added foreign keys to parser.py | 2022-03-10 |
| **Johan** | Added all the queries in the report and finished the report | 2022-03-10 |
| **Anas** | Finished GUI | 2022-03-10 |
| **Anas** | Fixing bugs due to formatting | 2022-03-12 |
| **Anas** | Deleting unused code and optimization | 2022-03-13 |
| **Anas** | Fixed video | 2022-03-14 |
| **Anas** | Fixed Readme | 2022-03-14 |
| **Johan** | Created view in database and added it to report | 2022-03-14 |