Assignment 3

Sarpreet Singh Buttar <sb223ce@student.lnu.se>
Songho Lee <sl222xk@student.lnu.se>

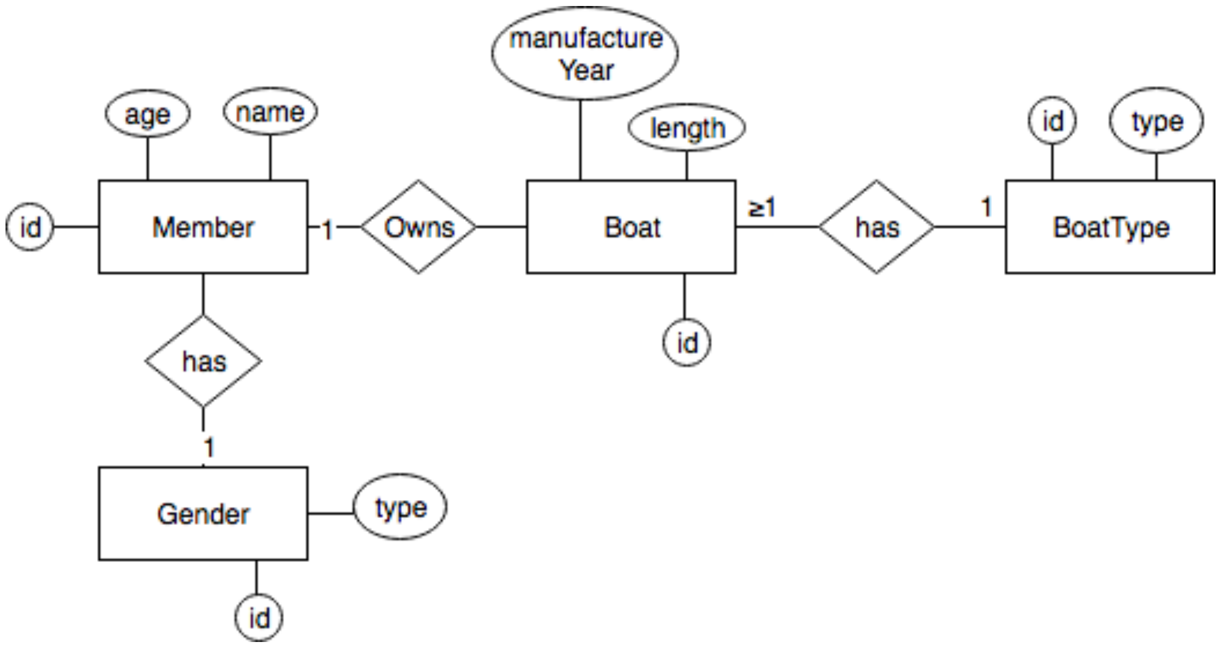
For the third assignment, we chose to create a management system for a boat club. The intended user of our programme is an administrator who wishes to manage a list of members and boats of a boat club. Our solution is suitable for a manager of a club as we provide a user-friendly web interface to add and edit members and boats. Besides, the user can search members and boasts in certain criteria. Therefore, the solution fulfils following:

1. Requirements

1. An admin can perform CRUD functionalities.
2. An admin can also perform search such as
    a. Get all members
        i. Get members by name
        ii. Get members by gender
        iii. Get members by age [≤, ≥, =, >, <]
    b. Get all boats
        i. Get all boats by length [≤, ≥, =, >, <]
        ii. Get all boats by manufacture year.
        iii. Get all boats by its type.
    c. Get all boats of a specific member
    d. Get a member from a specific boat.

We chose to use MySQL as a database management system, and programming language as JavaScript with React. You can find instructions on how to execute the project programme in a readme file.

2.Logical Model



Entities and Attributes

A database requires two major entities which are Member and Boat, as we are developing the system for these. We decided to include age, name attributes to identify a member, and for boats, we have its manufacture year, length as attributes.

We could treat gender of a member and type of a boat as attributes as well, but we decided to have these as separate entities. For boat type, we want to manage various sort of boats and having it as a separate entity eases updating and removing it, which several boats may already belong to. In case of the gender of a member, it would normally be treated in binary or short string form such as F/M. However, having it as a separate entity provides more flexibility of managing diverse type, such as gender identity. Furthermore, making these in entity sets makes the model fulfil normalised form

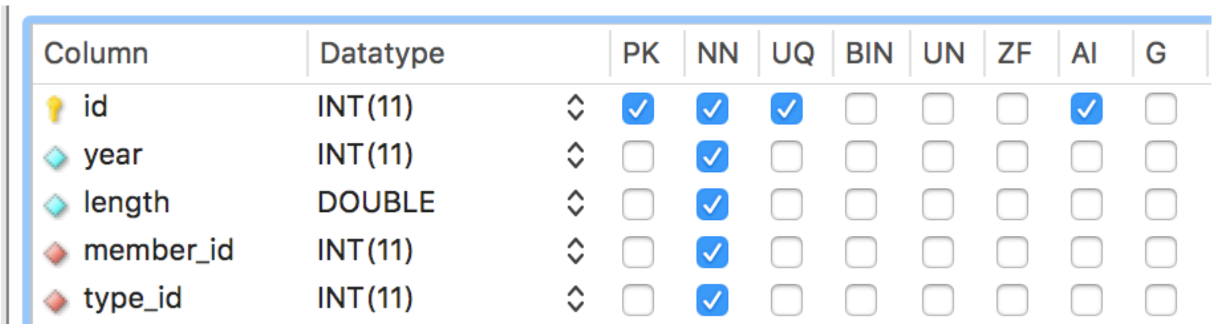
Relations

A single member can have many boats, but one boat is only allowed to have one owner in our design. A boat only has one boat type, but each boat types have various boats. A member has only one gender, and cannot belong to different genders simultaneously. If that is the case, the new type of gender can be defined instead. Several members can belong to a gender type.

3.SQL Design

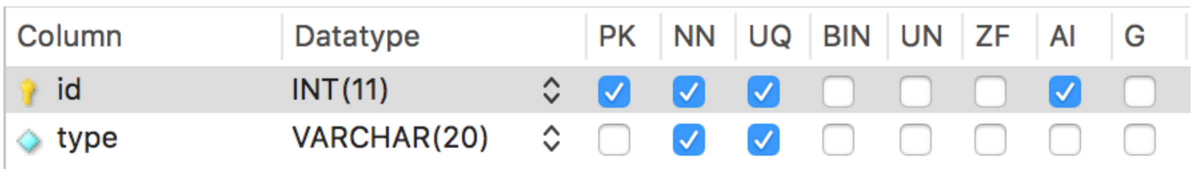
We translate our E/R diagram in following tables. We denote primary keys in underlined form and foreign keys with asterisk mark. Attributes (columns) are integer if these are not mentioned.

```
member(id, name, age, gender)
gender(id, type:string)
boat(id, year, length:double, member_id*, type_id*)
boat_type(id, type:string)
```



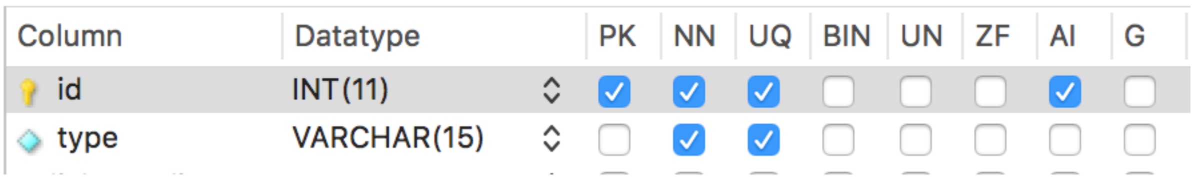
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
🔑 id	INT(11)	✓	✓	✓				✓	
◇ year	INT(11)		✓						
◇ length	DOUBLE		✓						
◆ member_id	INT(11)		✓						
◆ type_id	INT(11)		✓						

Figure 1 Boat table



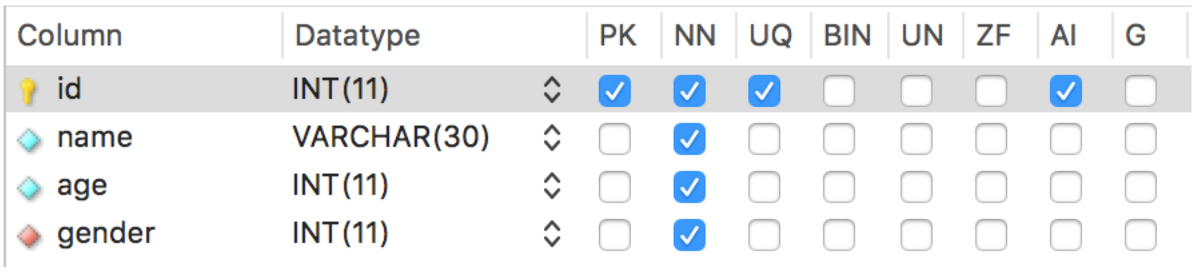
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
🔑 id	INT(11)	✓	✓	✓				✓	
◇ type	VARCHAR(20)		✓	✓					

Figure 2 Boat type table



Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
🔑 id	INT(11)	✓	✓	✓				✓	
◇ type	VARCHAR(15)		✓	✓					

Figure 3 Gender table



Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G
🔑 id	INT(11)	✓	✓	✓				✓	
◇ name	VARCHAR(30)		✓						
◇ age	INT(11)		✓						
◆ gender	INT(11)		✓						

Figure 4 Member table

The red square indicates that these are foreign keys. We applied to cascade on updates and restrict on deletion.

4.SQL Queries

Our programme utilises several queries to visualise entities from different tables into one table. Also, for our CRUD functionalities, we have queries to update and delete a row of a table. Here are several examples.

Queries on Boats

```
SELECT boat.id,boat.year,boat.length,member.id as memberId,
member.name as owner,boat_type.type
FROM boat_club.boat
JOIN member ON boat.member_id = member.id
JOIN boat_type ON boat.type_id = boat_type.id
```

The above query fetches a list of all boats from a boat table. As the boat table associates a member and type of a boat using their identifiers, we need to join these tables when we want to present all information in one table.

When we are fetching the list of boats upon a specific criterion, we append where clause to the above query. Examples follow.

```
WHERE member_id = ?
WHERE length > ?
```

CRUD functionalities of boats use following queries to create, modify and delete a boat.

```
INSERT INTO boat_club.boat SET ?
UPDATE boat_club.boat SET year = ?, length = ?, type_id = ? WHERE id
= ?
DELETE FROM boat_club.boat WHERE id = ?
```

Queries on Members

```
SELECT member.id, name, age, gender.type AS gender
FROM boat_club.member
JOIN gender ON member.gender = gender.id
```

This query fetches the list of members including their gender. Since we stored gender in a separate table, we are joining gender labels using the gender id. We rename the labels of the column as our programme is fetching the value using the alias.

Similar to searching boats, we append where clause after the above query to serach on members.

```
WHERE member.id = ?
WHERE gender = ?
```

Queries to create, modify and remove members looks as follows:

```
INSERT INTO boat_club.member SET ?
UPDATE boat_club.member SET name = ?, age = ?, gender = ? WHERE id = ?
DELETE FROM boat_club.member WHERE id = ?
```

5.Implementation

The programme is composed of two parts, one is a server which provides RESTful APIs, and the other is client part. The client programme provides a web interface.
These are written in JavaScript with React, and detailed instruction is in a readme file.

If you have a Node.js installed in your environment, executing '`npm start`' and '`npm run client`' on the 'boat_club' directory will make the service available. Former command is to run a server, and the latter is to operate the client side. The webpage is accessible on localhost:4000.

The programme makes a connection to the database in a remote location, which is reached at `songho.se:3306`. In case the database server is not available, you may find the database dump file we attach in this submission. If you are running MySQL server on your environment, you may visit /db/db.js and change the host, port, credential information and schema name of your database after importing the SQL dump file to your SQL server. But we hope the server is running so that this step is not necessary to follow. ☺