

# Database Report



UNIVERSITY  
*of*  
LIMERICK  
OLLSCOIL LUIMNIGH

Shelley Howarth - 17225663

Aaron Long - 17222478

Áine Reynolds - 17231515

Conor Canton - 16164571

# Table of Contents

<b>Work log</b>	<b>2</b>
<b>Database for Kayak club</b>	<b>3</b>
<b>Entity- relationship Diagram</b>	<b>3</b>
<b>Examples of tables</b>	<b>4</b>
Equipment	4
Kayaks	4
Oars	5
Life jackets	5
Members	5
Rented equipment log	6
<b>Functional dependencies</b>	<b>6</b>
<b>Third Normal form - 3NF</b>	<b>7</b>
<b>Views</b>	<b>8</b>
<b>Justification for triggers</b>	<b>10</b>

## Work log

17231515 - created schema.sql, report = 25%

17222478 - procedure, function and triggers = 25%

16164571 - views.sql, report= 25%

17225663 - ER diagram, functional dependencies and table proofs. = 25%

## Database for Kayak club

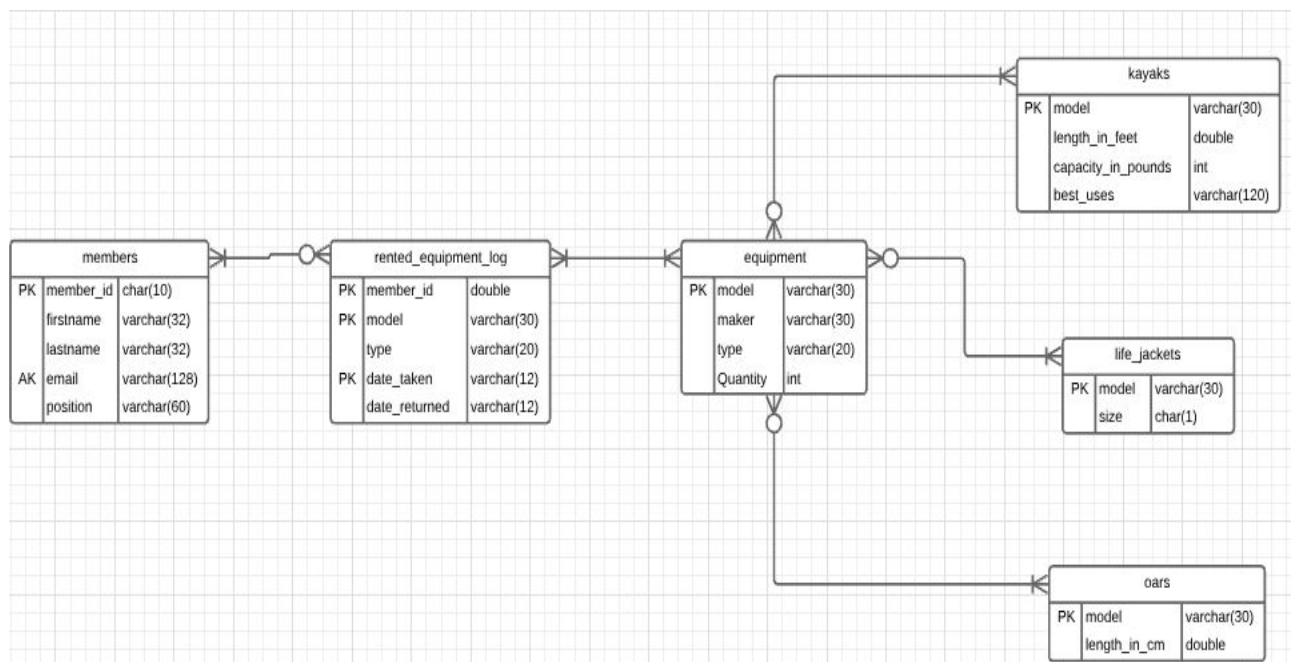
We decided to design a database for a kayaking club so they could break up what equipment they had and the various models for each type but also use it to keep a log of all the members that took out a piece of equipment.

Our reasoning behind this database was to allow a simple club such as a kayaking club to have more structure. The database would also give them the opportunity to keep track of all their equipment. If something went missing or a piece of equipment was damaged they would be able to look back at the log and see which member was last to have that particular piece of equipment.

We broke the equipment up into kayaks, oars and life jackets because have them in separate tables allows for extra information to be displayed which could be helpful for when people are trying to decide which model best suits them and what they are going to do.

Our software system would be for the committee members of the club to use so they can keep track of things more efficiently. All the views that we've created would be options that the software user can choose to do, such as viewing all the members who have yet to return a piece of equipment. The database would be incorporated into the software system so the views can refer back to it.

## Entity- relationship Diagram



# Examples of tables

## Equipment

+ Options

  Edit  Copy  Delete	model	maker	type	quantity
  Edit  Copy  Delete	1001	Wilderness	oar	14
  Edit  Copy  Delete	1002	EddyLine	oar	20
  Edit  Copy  Delete	1003	Jackson	oar	10
  Edit  Copy  Delete	1004	Native Watercraft	oar	30
  Edit  Copy  Delete	1005	Bonafide	oar	22
  Edit  Copy  Delete	1006	EddyLine	oar	12
  Edit  Copy  Delete	1007	FeelFree	life jacket	20
  Edit  Copy  Delete	1008	Native Watercraft	life jacket	14
  Edit  Copy  Delete	1009	OldTown	life jacket	25
  Edit  Copy  Delete	1010	Jackson	life jacket	19
  Edit  Copy  Delete	1011	Bonafide	life jacket	21
  Edit  Copy  Delete	1012	Bonafide	life jacket	23
  Edit  Copy  Delete	Adaro	Bonafide	kayak	3
  Edit  Copy  Delete	Kelpie	FeelFree	kayak	7
  Edit  Copy  Delete	Kurki	NuCanoe	kayak	18
  Edit  Copy  Delete	Mora MV	Jackson	kayak	10
  Edit  Copy  Delete	Selki HV	NuCanoe	kayak	14

Primary key: model

## Kayaks



















+ Options

  Edit  Copy  Delete	model	length_in_feet	capacity_in_pounds	best_uses
  Edit  Copy  Delete	Adaro	12.6	250	recreational
  Edit  Copy  Delete	Kelpie	15.9	190	small paddlers
  Edit  Copy  Delete	Kurki	17.3	240	windy conditions - ocean
  Edit  Copy  Delete	Mora MV	15.6	220	rock gardening, ocean, lake
  Edit  Copy  Delete	Selki HV	15.9	280	fishing, ocean, lake

Primary key: model

## Oars



















+ Options

						model	length_in_cm	
<input type="checkbox"/>		Edit		Copy		Delete	1001	230
<input type="checkbox"/>		Edit		Copy		Delete	1002	240
<input type="checkbox"/>		Edit		Copy		Delete	1003	210
<input type="checkbox"/>		Edit		Copy		Delete	1004	250
<input type="checkbox"/>		Edit		Copy		Delete	1005	220
<input type="checkbox"/>		Edit		Copy		Delete	1006	260

**Primary key: model**

## Life jackets
















+ Options

					model	size		
<input type="checkbox"/>		Edit		Copy		Delete	1007	M
<input type="checkbox"/>		Edit		Copy		Delete	1008	S
<input type="checkbox"/>		Edit		Copy		Delete	1009	L
<input type="checkbox"/>		Edit		Copy		Delete	1010	M
<input type="checkbox"/>		Edit		Copy		Delete	1011	S
<input type="checkbox"/>		Edit		Copy		Delete	1012	L

**Primary key: model**

























## Members

+ Options

					member_id	firstname	lastname	email	position		
<input type="checkbox"/>		Edit		Copy		Delete	123343	Sarah	Jones	123343@ul.ie	Chairperson
<input type="checkbox"/>		Edit		Copy		Delete	154423	Dan	Fitz	154423@ul.ie	Committee member
<input type="checkbox"/>		Edit		Copy		Delete	17232	Anne	O'Brien	17232@ul.ie	Basic member
<input type="checkbox"/>		Edit		Copy		Delete	173455	Jack	Morrissey	173455@ul.ie	Basic member
<input type="checkbox"/>		Edit		Copy		Delete	18123	Aisling	Dean	18123@ul.ie	Basic member

**Primary key: member\_id**

## Rented equipment log

+ Options				member_id	model	type	date_taken	date_returned
<input type="checkbox"/>	 Edit	 Copy	 Delete	17232	1012	life jacket	2017-03-12	2017-03-15
<input type="checkbox"/>	 Edit	 Copy	 Delete	18123	Mora MV	kayak	2017-05-22	2017-04-25
<input type="checkbox"/>	 Edit	 Copy	 Delete	18123	FeelFree	kayak	2018-05-22	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	18123	1001	oars	2018-11-20	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	18123	1002	oar	2018-11-24	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	123343	1011	life jacket	2018-01-20	2018-01-22
<input type="checkbox"/>	 Edit	 Copy	 Delete	154423	1006	oar	2017-05-30	2017-06-02
<input type="checkbox"/>	 Edit	 Copy	 Delete	173455	Adaro	kayak	2018-01-09	2018-01-10

**Primary key: member\_id and date\_taken**

## Functional dependencies

### Members

Member_id	→	firstname
Member_id	→	lastname
Member_id	→	email
Member_id	→	position

### Kayaks

Model	→	Length_in_feet
Model	→	Capacity
Model	→	Best use

### Oars

Model	→	Length_in_cm
-------	---	--------------

### Life Jackets

Model → Size

### Equipment

Model → Maker

Model → Type

Model → Quantity

### Rented Equipment Log

Member\_id, date\_taken → Type

Member\_id, date\_taken → Date returned

## Third Normal form - 3NF

### Members

The table “members” is in third normal form as all the attributes that are not primary keys depend on the primary key of “member\_id” in this case. For our tables to be in third normal form we must satisfy the rules for 1NF, 2NF and of course 3NF. Each table must not have a duplicate piece of data within the same single row. Any data that is not directly related to the key of the table must be removed to prevent adding anything unnecessary to the tables. This is important in “members” as you do not want a member to be added to the table twice.

### Kayaks

We can prove that the kayaks table is in third normal form due to the fact that all the attributes depend on the primary key which is “model”. In the table “kayaks”, the same model of the kayak is not needed twice. Having the primary key as model makes sure we can’t have a model in more than one column. This again proves the third normal form format of the table.

### Oars

In the “oars” table the primary key is “model”, the other attribute in the oars table is “length\_in\_cm”, the value of which is dependent on the model. The table satisfies the rules of first and second normal form, it now therefore also satisfies third normal form.

## Life Jackets

The life jackets table is in third normal form due to the fact that all the attributes other than the primary key, depend on that primary key. In this case the primary key is “model”, the model of a life jacket can be used to determine the size of that life jacket.

## Equipment

As in the other tables, the table “equipment” is in third normal form as it satisfies the conditions of first and second normal form. We can again see that the primary key is “model” and is the factor that the other attributes of type, maker and quantity depend on. If the model was to change, none of the other attributes would be correct.

## Rented Equipment log

We can see that the rented equipment log is in third normal form type and date returned are functionally dependent on “member\_id”, “model” and “date\_taken”. This table also has no transitive dependencies which means that the table rented equipment log is in third normal form.

# Views

Several views were proposed and designed for possible use in an imaginary software system. These views were designed to help the user easily access certain information.

## Members/Committee List

Here the user can select to see the list of members in the club. They can select to view a regular member list showing the member’s names or a committee list showing the position and name of committee members.

## Oar/Kayak/LifeJacket List

These views allow for the user to view the current stock and inventory of equipment which the club has. It combines all the sizes and makes together. This allows for easy management of the equipment for the equipment manager and ensures no equipment goes missing. We used ROLLUP to display the total quantity in the life jacket list but were unable to get the text “Total” in the same row.



## Members email list

The user can select this view to create an email list for the club. It allows a quick and easy list to be created. This can be used for the public relations officer who may need to send emails to all members in the club or to a specific member.

## Rented Equipment 2017/2018

This shows a list of all the equipment that was rented in the year selected. It provides the user with the name, date rented and what piece of equipment was rented. It is used to help keep track of who has what piece of equipment and to ensure all rented equipment is returned.

## Rented Equipment to be returned

This is similar to view 4 but it shows the list of people who have taken out equipment but have yet to return it. It provides the email of the member as well as their name which allows for the member to be easily messaged to enquire about when the gear is been returned. This helps equipment been robbed or taken by members who don't have authorisation.

## Non active members

This view is used to show the members who haven't rented equipment in 2018 but still are members. This view can be used to see how many active members there are in the club. Active members are classed as people who have rented equipment in the past year.

## Indexes

We used the indexes of the primary key for all the tables and the unique key (email) in the table members. The good thing about having these indexes is that they can speed up the time taken to carry out a query. These primary and unique keys are stored in B-Trees which have enough space to store a number of key pointer pairs. Having the primary key as our index allowed us to efficiently run our view queries, with none of them taking more than 0.001 of a second.

View Name	Avg View Speed(s)
committee_list	0.0004
kayak_list	0.0008
life_jacket_list	0.0006
oar_list	0.0007
Members_list	0.0006

member_email_list	0.0004
rented_equipment_to_be_returned	0.0005
rented_equipment_2017	0.0007
rented_equipment_2018	0.0007
Non_active_members_2018	0.0008

## Justification for triggers

### Triggers

For this area of the project we created two triggers, one that would delete a member from “rented\_equipment\_log” if the same member had been deleted in “members” and another that would update the quantity of the equipment in the table “equipment”. We found that both these triggers would help improve the system by making sure that the tables stay up-to-date with the various changes that could occur in a kayaking club.

#### UpdateRented

The “UpdateRented” trigger was implemented to remove any members from the table “rented\_equipment\_log” after a member has been deleted from “members”. The trigger also implements the “DoesMemberExist” function which returns an integer that checks if there is a member exists in “rented\_equipment\_log” but not in “members”. Depending on what integer is returned, the trigger will delete the member in “rented\_equipment\_log” that is not in “members”

#### UpdateQuantity

The “UpdateQuantity” trigger automatically decrement the equipment of a specific model by 1 in the table “equipment” if a member has taken out a piece of equipment to rent. The trigger is executed when the “AddLoan” procedure is called, it finds the model number of the piece of equipment that the member rented out in the table “equipment” and proceeds to decrement this quantity by 1.

### Stored Procedure

Procedures are essential to some database systems due to the reusability of each one. We created three procedures in our system to make it more efficient. These three procedures are called AddLoan, DeleteMember and AddMember. We decided on these procedures as they would be the ones most often used in club.

#### AddLoan

The AddLoan procedure is based on the renting of equipment within the club. This procedure will give the admin the ability to log the member who has loaned out the equipment to the rented\_equipment\_log, along with the model, type of equipment and the date it was taken out. It will not include “date\_returned” because the item has yet to be returned.

## DeleteMember

The DeleteMember procedure allows us to delete a member from the members table. This procedure is linked with the trigger "UpdateRented", so when the procedure is called the trigger is automatically executed.

## AddMember

The "AddMember" procedure accounts for the adding of the member to the club. It requests an input of five values (member ID, first name, last name, email address and position) that creates a member in the table "members". These values that are being taken in will be added to the table in respective order. The member can vary in position within the club, ranging from the regular member to chairperson.

## Functions

A function is a programming construct that accepts parameters, does work that typically makes use of the accepted parameters, and returns a type of result. They're used more for reading data rather than modifying data. You can also define a function as a group of SQL statements that would perform a specific task. In our schema we have created only one function. This function is called "DoesMemberExist".

## DoesMemberExist

The "DoesMemberExist" function checks if a member exists in the table "rented\_equipment\_log" but not in the table "members". The function returns an integer. Within this function we have two checks to see if the member either exists in the members table or in the rented\_equipment\_log table. If either table doesn't contain the member id then it will return the integer 1. If it returns 0 it means that there is a member that exists in "rented\_equipment\_log" but not "members". Since this function is implemented in the "UpdateRented" trigger, the trigger will delete the row containing the member in "rented\_equipment\_log" when the function returns 1.