Database Prototyping Games Rental System

# By Craig Banyard

# Jordan Fulton

# Ben Moore

# Joseph Kellaway

# And Dan Scott

# Ismini Vasileiou

# University of Plymouth

# 11/01/2016

|  |  |  |
| --- | --- | --- |
| 1 | Introduction | 2 |
| 2 | Functional Requirements | 2 |
| 3 | Testing | 5 |
| 4 | Conclusion | 5 |
| 5 | Appendix 1 | 6 |
| 6 | Appendix 2 | 7 |
| 7 | Appendix 3 | 15 |

# Introduction

Our objective is to understand the current system and redesign it while eliminating its weaknesses. To achieve this, we listed every entity which was used in the current system plus any data we thought the user might need. We drew an ERD using the sample data deriving what we thought would be entities. After a few revisions we had what we agreed was the correct ERD for an efficient system and then we started the normalisation. Once the normalisation was complete we started building the tables and then using our ERD, looked into the defining the relationships between tables. This took a few revisions to get right, but after we all agreed we had found the correct series of relationships we started to build the rest of the application within Access.

Our new system tried to improve upon the old system by having a normalised backend which eliminated the redundancy within the data. We also wanted the GUI to be user friendly and make it easy for the user to query the data as well displaying specific data in an easy to read manner, instead of overloading the user with lots of information at once. To achieve this we made sure every window had a specific function and gave a main menu to the user with a series of buttons. Each opening a window for a specific function.

*For Project Plan please see Appendix 1.*

*For Data Modelling see Appendix 2.*

# Functional Requirements

## Add/Amend a Member

To implement this feature, we added a form to update the members table which holds all members details. We created a form to search and amend existing members as well as creating new ones. This would either add or update records in the members table accordingly. The form that updated this information was designed to hold all the information and be quick and easy to use as well as easy to search by, allowing the user to do a partial search by either first name, surname or member ID number so that the user could easily find required information.

## Add/Amend a Game/Game Copy

We also added a form which allows us to add games into the system. When adding a game, we designed the system so you have to add at least one copy of each game. When a game title is added to the system this means that there will not be any games entered into the system without copies. We can also add multiple copies of the game when adding a game, you can also set the price the user paid for the game initially and the supplier for each game copy. This means the user only has to enter the title, rental rate and developer once. This makes the system less prone to error as information is only entered once.

## Issue a Rental/ Renew a Rental

Once we could add and remove users, games and game copies, we implemented the ability to add a rental. This was done by populating the date and member ID fields within the game copy table. When a user wants to create a rental the user will search for a member by member ID and this will bring up all relevant information for the user as well as all their current rentals. Then the user will search for the game to see if there are any copies in stock. When searching for game copies to add to the rental, the system will only show game copies that are not currently associated with a member ID which indicates it is not being rented. Once that is all selected, that member is assigned that game copy for the duration of the rental.

Renewing a rental works in a very similar way because it just extends the current rental’s return date by up to two weeks. This is not possible if there is a current reservation for the game.

## Return a Game Copy

When returning a game, the system copies the rental information from the game copy table over to the rental history table. This is then purged from the game copy table which makes the game copy visible again when the user searches for it on the system to create a new rental. To make this simple for the user it is just a button that says return. This opens a form displaying the rental duration, the number of days the rental is overdue and how much to charge the customer. It also has an editable notes field to allow the user to comment on the condition that the game has been returned in. The expected return date is used for the cost of the rental if it is returned early, we have left it at the discretion of the user whether to discount this.

## Manage an overdue Game Copy Rental

The system compares the expected return date field to today’s date so that the user can see that the game is overdue without actually checking it. The system also blocks the user from making any new rentals, reservations or renting any current reservations until the overdue game copy is returned. The additional cost of the overdue rental is also calculated for the user when it is returned. The number of concurrent rentals can be altered at the user’s discretion to deal with members that regularly keep overdue rentals.

## Reserve a Game Copy

Once we had completed the core functionality we began creating additional features like the reservation system. To implement this we created a new table to hold reservations. When creating a reservation, the application creates a new record in this table which holds the catalogue ID of the game, member ID and the requested date. The reservation is made for the game and not a specific copy since there are factors that could prevent the copy from ever being returned in a usable state. Reservations are made in the same way as rentals, except they are made when the game has no copies in stock. We added two sub-forms to the rental form to display reservations that have been made and another that populates are reservations that have been made are back in stock to ensure that it is simple to use.

## Create a Rental History Record

The user will never directly add to the table as it is populated when the system is told that a rental has been returned but can view it in reports generated by various queries. It holds the stock and member ID numbers, issue date, expected and actual return dates, a calculated field that works out how many days (if any) the item is overdue and whether the customer paid or not.

## Add/Amend Rental Rate

We have assumed that there will be tiers of rental rates where games over a certain age drop in value and therefore have a reduced rental rate. This would mean that editing each game’s cost individually would be mundane and inefficient. The user can create new and edit existing rental rates from relevant forms that we have provided as well as change a single game’s rental rate ID (and thus changing its rental rate) from the edit game form. The user would have the option of moving games between different rates individually or simply keeping games with similar release dates on the same rental rate ID and editing the daily charge value of that rental rate.

## Add/Amend a Supplier

Inputting supplier information for every single game purchased would be extremely time consuming and inefficient. We have a table set up to give each supplier an ID and store the relevant contact information there as well as a notes section in the event that the user decides to not use a supplier again for a specific reason. This table is added to and edited using two separate forms in the same way that the other add and edit forms work to ensure ease of use. The supplier ID is therefore all that is needed in the game table meaning we managed to cut down on a lot of data redundancy found within the old system.

## Search Catalogue Based on Specific Criteria

The next feature we decided to implement within our application was the search feature of the catalogue, this is designed for members coming into the shop who want to search the entire catalogue of the shop. We felt that the best way to supply this would be to allow the user to query the database using a form and produce a report from the query. We developed queries which searched for all the information only the user needed so things similar to Category Name, Title, Platform Age Rating, Reviews and Rental Cost etc. Things that the members would need to see before deciding to rent a game, but not things that they don’t need to see like the supplier or how much the shop paid for the game.

## View Games

We also developed a report for viewing games which is designed to be used by the staff at the shop. This report works in the same way as the catalogue but displays different information because the manager or shop assistant aren’t likely to want to see reviews of games or customer ratings of games in this way.

## View/Report Rental History

For the shop manager we developed a series of queries for creating reports that show relevant rental information (member details, the game details and rental dates as well as renewals). Whilst they requested rental history we felt that simply displaying all historical rentals at once would be of limited use. We have allowed the user to search for all current and historic rentals as well as all rentals within two input dates. We have also allowed the user to narrow this search by searching current, historic and rentals within those two dates by the title of the game.

## View/Report Members

Like the rental history report it is feasible that the manager needs to view a list of all members and information stored on them. We have a form with buttons to generate a report of all members as well as giving the option to narrow this down to all members with a current rental. We have also added the option to search for all members in a specific town or county as we felt that if the company wished to expand or move premises then knowing where most customers are would be useful.

# Testing

We covered most of our testing as we carried out our implementation – we would be building forms, queries, adding buttons etc. and using them immediately to ensure that they work as intended but we did draw up a test table in when planning out the project so we did a final check and filled it out. We did discover some issues that we had missed or created adding extra functionality so it was useful in the end.

*See Appendix 3 for test table.*

# Conclusion

In conclusion we believe that we have vastly improved on the original system. We have added more advanced features and have made the system more user friendly and definitely easier to use. If we were to do this again then we would add some extra features to the system, for example, a login system. This way different members of staff have different levels of access to the system and it would be possible to leave a terminal running to allow customers to access the catalogue. Also allowing the customer to reserve games from the catalogue would be another function we’d like to add – pending the user’s approval. We feel that this would make the shop far more efficient and it would allow the members to be able to make reservations by themselves rather than relying on the shop staff.

Looking back at our work we have also realised that we could have implemented age rating and platform fields as their own tables in the same way that we did with the rental rate and supplier information. This would cut down on data redundancy within the database further and thus make it a slightly more efficient solution.

Overall as a group we think we have worked a lot on this system and are very proud of what we have made. We believe that the project management was handled well and shared around equally during the time of this project and we feel as if the end product meets the requirements of the user.

# Appendix 1

# Assignment Plan

## Week beginning 23rd November 2015

Analyse and produce ERD and Normalise the sample data inputs. Complete the assignment plan. Contact client to review and give feedback on ERD and Normalisation of sample data inputs. Amend any changes discussed with the client.

## Week beginning 30th November 2015

Create a database with the following functionality:

* Add / Amend a Member
* Add / Amend a Game
* Add / Amend a Game Copy
* Issue a Game Copy Rental
* Return a Game Copy Rental

Discuss implementation with Client and amend any issues the client raises.

### Client feedback

The client prefers each form to have a single role – it should not be possible to edit existing games on the add game form for example.

Supplier: company that sells games i.e. Amazon, Game etc.

Distribution Company: company that creates games i.e. Sega, Activision etc.

## Week beginning 7th December 2015

Add the following functionality to the database:

* Manage an overdue Game Copy Rental
* Reserve a Game Copy
* Cancel / Delete a reservation
* Create a rental history record
* Add / Amend Catalogue information (such as a review or screenshot)
* Add / Amend a Rental Rate
* Add / Amend a Distribution company
* Add / Amend a Supplier
* Search catalogue on criteria such as Game, Genre and Age rating
* View / Report Game and / or Game Copies
* View / Report Rental(s)
* View / Report Rental History
* View / Report Members
* View / Report Reservations
* View / Report Overdue Rentals
* View / Report Members Overdue
* View / Report Popular Games

Ask Client for feedback on additional functionality and amend accordingly.

### Client feedback

Currency in dollars – please change.

Customer pays entire bill on return of game including any renewals and overdue costs.

## Week beginning 14th December 2015

Produce report and start to produce user guide.

### Client feedback

Client appeared content with our product.

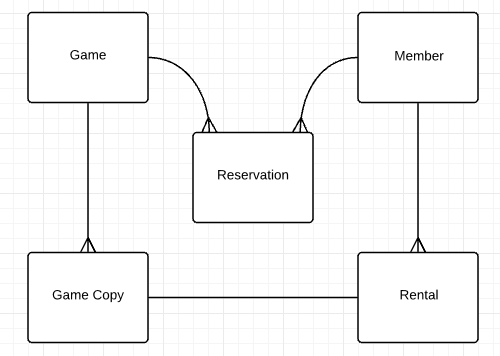
## Week beginning 21st December 2015 to week ending the 9th January 2016

To finalise user guide, run test data on the database and refine final product.

# Appendix 2

# Data Modelling

## Initial Entity Relationship Diagram



## Data Normalisation

We split the data normalisation into parts so that it would be easier to implement the database bit by bit. First we normalised the data relevant to the initial ERD:



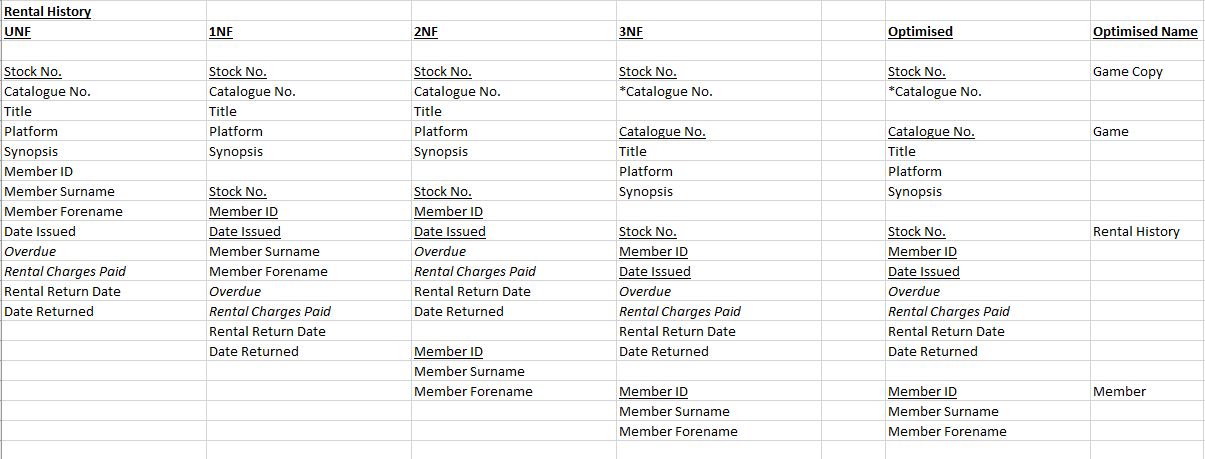




Which merged into the following tables:



We then added rental history data:



And finally the information required for the catalogue that the customer can look through:



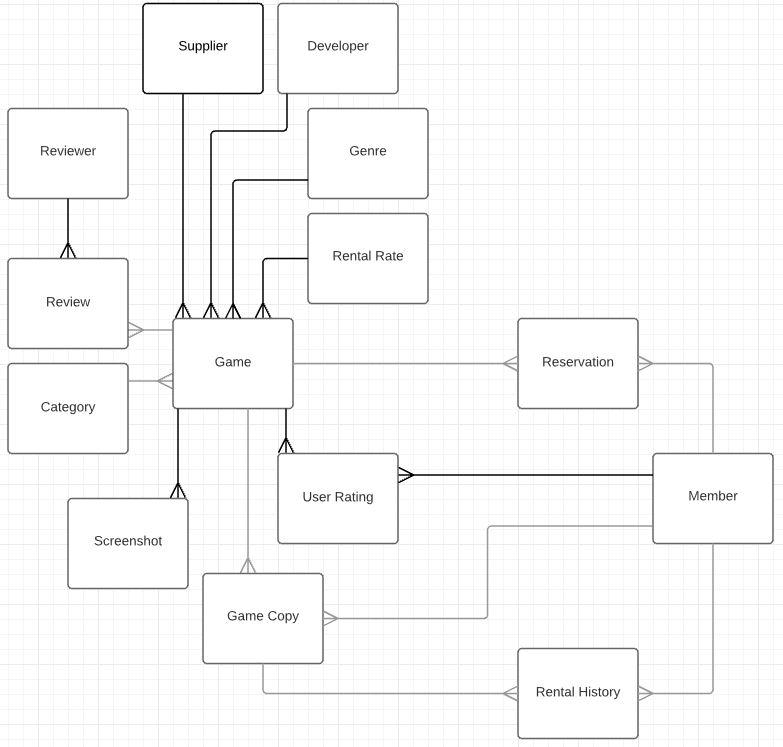
This left us with the following merged tables in our final product:







## Final Entity Relationship Diagram



# Appendix 3

# Test Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item to Test | Test Description | Test Date | Outcome | Action |
| Add/Amend a Member | Add a member from the form.  Amend a member from the form. | 18/12/2015 | Works.  Works. | Add input masking.  Fixed. |
| Add/Amend a Game | Add a game from the form.  Amend a game from the form. | 18/12/2015 | Works.  Works. | Fix tab ordering.  Fixed. |
| Add/Amend Game Copy | Add a game copy from the sub-form. | 18/12/2015 | Works.  Works. | Ensure that you must add game copy when you add a game.  Fixed. |
| Issue a Game Copy rental | Make a rental through the forms. | 18/12/2015 | Works. | None. |
| Renew a Game Copy rental | Press the renew button and input a new return date. | 18/12/2015 | Adds details to rental history but doesn’t renew loan. | Fix.  Fixed. |
| Return a Game Copy rental | Press the return button and choose whether the rental is paid for or not. | 18/12/2015 | Works. | None. |
| Manage an overdue Game Copy rental | Ensure that the rental is billed correctly. Ensure that member cannot make another rental. | 18/12/2015 | Works. | None. |
| Reserve a Game Copy | Attempt to rent a game with no copies in stock. | 18/12/2015 | Works. | None. |
| Cancel/delete a Reservation | Attempt to cancel a reservation. | 18/12/2015 | Works. | None. |
| Create a Rental History record | When rental is returned check the rental history table is populated. | 18/12/2015 | Works. | None. |
| Add/amend Catalogue information | Check forms to add screenshots, reviews and reviewers. | 18/12/2015 | Works. | None. |
| Add/Amend a Rental Rate | Test add and amend rental rate forms. | 18/12/2015 | Works. | None. |
| Add/Amend a Developer | Test add and amend developer forms. | 18/12/2015 | Works. | None. |
| Add/Amend a Supplier | Test add and amend supplier forms. | 18/12/2015 | Works. | None. |
| Search Catalogue based on criteria | Test catalogue search form. | 18/12/2015 | Works. | None. |
| View Games and/or Game Copies | Test game and game copy reports. | 18/12/2015 | Works. | None. |
| View/Report Rentals | Test rental report. | 18/12/2015 | Works. | None. |
| View/Report Rental History | Test rental history report. | 18/12/2015 | Works. | None. |
| View/Report Members | Test members report. | 18/12/2015 | Works. | None. |
| View/Report Reservations | Test reservations report. | 18/12/2015 | Works. | None. |
| View/Report overdue Rentals | Test overdue rentals report. | 18/12/2015 | Works. | None. |
| View/Report Members overdue | Test members with overdue rentals report. | 18/12/2015 | Works. | None. | |
| Report popular Games | Test popular games report. | 18/12/2015 | Works. | None. | |