**Simple Database**

Table of Contents:

[Introduction 1](#_Toc75362683)

[Normalization 1](#_Toc75362684)

[Model of Data 3](#_Toc75362685)

[Implementation 3](#_Toc75362686)

[Raw Code 5](#_Toc75362687)

[Conclusion 6](#_Toc75362688)

[References 6](#_Toc75362689)

# Introduction

The anime industry is one of, if not, the biggest genres in the entertainment industry. There are tens of thousands of different anime with many more being made every season. I have chosen a data set on anime which represents just a small amount of this vast genre, but more can be added I just used a small sample amount of data in order to create a functioning, efficient database. In this report I will create a data model and implement a data base that will reduce the amount of storage and unnecessary data from the original data set to optimise efficiency and only keep what is needed. The database will represent users inputting their rating for a given anime.

# Normalization

Normalization is a technique used in database design, which can be used in order to design relational databases to higher normal forms. The main goal for using normalization is to “reduce data redundancy and maintain the atomicity within the database table” (K. Kumar and S. K. Azad 2017). For normalization in the first normal form each cell needs to be single valued, entries in a column must be the same type and each row needs to be uniquely identified using unique ID’s or adding more columns to make it unique. In the table below ‘Genre’ has more than one value so in order to make it in first normal form another table needs to be created where the genres are in separate entries. **Figure 1** shows a sample of the main dataset being converted ( ‘’ in a cell refers to data not changing from above values). Highlighted value is the primary key of the table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User\_name | User\_id | Anime\_id | Name | Genre | Type | Episodes | Rating | members |
| Mayyhin | 1 | 32281 | Kimi No Nawa | Drama, Romance, School, Supernatural | Movie | 1 | 9.37 | 200630 |
| Alan | 2 | 12355 | Ookami Kodomo no Ame to Yuki | Fantasy, Slice of Life | Movie | 1 | 8.84 | 226193 |
| Josh | 3 | 2001 | Tengen Toppa Gurren Lagann | Action, Adventure, Comedy, Mecha, Sci-Fi | TV | 27 | 8.78 | 562962 |
| Adam | 4 | 19 | Monster | Drama, Horror, Mystery, Police, Psychological, Seinen, Thriller | TV | 74 | 8.72 | 247562 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User\_name | User\_id | Anime\_id | Name | Genre | Type | Episodes | Rating | members |
| Mayyhin | 1 | 32281 | Kimi No Nawa | Drama | Movie | 1 | 9.37 | 200630 |
| ‘’ | ‘’ | ‘’ | ‘’ | Romance | ‘’ | ‘’ | ‘’ | ‘’ |
| ‘’ | ‘’ | ‘’ | ‘’ | School | ‘’ | ‘’ | ‘’ | ‘’ |
| ‘’ | ‘’ | ‘’ | ‘’ | Supernatural | ‘’ | ‘’ | ‘’ | ‘’ |

***Figure 1*** *– Dataset put into first normal form (1st normal form table only used data from one anime\_id as an example)*

Now that the able is in first normal form I need to convert it into second normal form. In order to do this each non-key value must be dependant on a primary key. For this, 2 more tables will need to be made; one with a primary key for User\_id and one with a primary key for Anime\_id. **Figure 2** shows the table in second normal form. This allows for a single user (user\_id) to have many different submissions (anime\_id) having a one-to-many relation to the database. (Highlighted columns are primary keys). User\_id is used as a primary key instead of user\_name as many people could potentially have the same username but the user\_id is unique to each user.

|  |  |
| --- | --- |
| User\_id | User\_name |
| 1 | Mayyhin |
| 2 | Alan |
| 3 | Josh |
| 4 | Adam |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Anime\_id | Name | Genre | Type | Episodes | Members |
| 32281 | Kimi No Nawa | Drama | Movie | 1 | 200630 |
| ‘’ | ‘’ | Romance | ‘’ | ‘’ | ‘’ |
| ‘’ | ‘’ | School | ‘’ | ‘’ | ‘’ |
| ‘’ | ‘’ | Supernatural | ‘’ | ‘’ | ‘’ |

|  |  |  |
| --- | --- | --- |
| user\_id | anime\_id | Rating |
| 1 | 32281 | 9.37 |
| 2 | 12355 | 8.84 |
| 3 | 2001 | 8.78 |
| 4 | 19 | 8.72 |

***Figure 2*** *– Dataset from* ***figure 1*** *converted into second normal form.*

I do not need to convert this into third normal form as the only necessary data is the user\_id, anime\_id and the rating the user gave that specific anime\_id. Putting it into third normal form would be overcomplicating it for no reason as it has all relevant tables for its given task, which is to allow users to rate anime they have watched.

# Model of Data

The diagram in **Figure 3**  shows a data model for anime ratings given to by users depending on the certain anime. The main table being the child table made from the two parent tables (user\_table and anime\_table). The child table having a one-to-many relationship with both parent tables as one user can give many anime ratings and one anime can have multiple ratings from different users (even though its not in the dataset, it is still a possibility). Due to the task being fairly simple the model is small and not complex as we do not need to create tables for sub-entities like genres and types of anime as it is unnecessary and a waste of storage.

A screenshot of a computer

Description automatically generated with medium confidence

***Figure 3 –*** *Anime Ratings data model diagram.*

# Implementation

Text

Description automatically generated

***Figure 4 –*** *MySQL code for anime\_rating table (main table).*

In **Figure 4** the first line of code ‘CREATE TABLE’ creates the table called ‘anime\_rating’ , this function will be used multiple times to create different tables in others figures shown. Lines 3-5 all define 3 attributes being user\_id, anime\_id and ratings. User\_id and anime\_id are foreign keys from the other 2 tables these are defined as primary keys between lines 7-11. These foreign keys are also the primary keys of this table.

Text

Description automatically generated

***Figure 5 –*** MySQL code for anime\_table.

Everything encased within the brackets different attributes of the database are defined. The attributes are defined after the brackets by adding the name of the attribute between apostrophes followed by the data type and whether it is null or not. These are all not null as at this current stage for the task I am using it for I do not want these values to change. On line 10 ‘anime\_id’ is defined as this tables primary key. Normally unique id’s would have an ‘AUTO\_INCREMENT’ function but this dataset already had unique ids for each anime so it was not needed.

Text

Description automatically generated with medium confidence

***Figure 6 –*** *MySQL code for user\_table.*

This SQL code in **Figure 6** creates a table for users who would be inputting their ratings on the anime of their choosing. Same as the table in **Figure 5,** but the primary key in this case is ‘user\_id’.

***Graphical user interface, text

Description automatically generated***

***Figure 7 –*** *MySQL code for inserting into the database.*

The SQL code in **figure 7**  shows how data is inserted into the anime\_table of the database. Full code for all the anime being inserted can be found in the **Raw Code** section.



***Figure 8 –*** *MySQL code for deleting*

The SQL code in **figure 8**  shows how a record from the table ‘anime\_table’ with the ‘anime\_id’ of ‘5114’ is deleted. Any anime\_id can be used this was just used as an example to show how it works.

# Raw Code

This section will just have some raw code if you are interested in seeing the raw code of how the data for anime\_tables was inserted.

INSERT INTO `anime\_table` (`anime\_id`, `name`, `genres`, `type`, `episodes`, `members`) VALUES ('32281', 'Kimi no Na wa.\r\n', 'Drama, Romance, School, Supernatural\r\n', 'Movie\r\n', '1', '200630'), ('5114', 'Fullmetal Alchemist: Brotherhood\r\n', 'Action, Adventure, Drama, Fantasy, Magic, Military, Shounen\r\n', 'TV\r\n', '64', '793665'), ('28977', 'GintamaÂ°\r\n', 'Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen\r\n', 'TV\r\n', '51', '114262'), ('9253', 'Steins;Gate\r\n', 'Sci-Fi, Thriller\r\n', 'TV\r\n', '24', '673572'), ('9969', 'Gintama&#039;\r\n', 'Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen\r\n', 'TV\r\n', '51', '151266'), ('32935', 'Haikyuu!!: Karasuno Koukou VS Shiratorizawa Gakuen Koukou\r\n', 'Comedy, Drama, School, Shounen, Sports\r\n', 'TV\r\n', '10', '93351'), ('11061', 'Hunter x Hunter (2011)\r\n', 'Action, Adventure, Shounen, Super Power\r\n', 'TV\r\n', '148', '425855'), ('820', 'Ginga Eiyuu Densetsu\r\n', 'Drama, Military, Sci-Fi, Space\r\n', 'OVA\r\n', '110', '80679'), ('15335', 'Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare\r\n', 'Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen\r\n', 'Movie\r\n', '1', '72534'), ('15417', 'Gintama&#039;: Enchousen\r\n', 'Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen\r\n', 'TV\r\n', '13', '81109'), ('4181', 'Clannad: After Story\r\n', 'Drama, Fantasy, Romance, Slice of Life, Supernatural\r\n', 'TV\r\n', '24', '456749'), ('28851', 'Koe no Katachi\r\n', 'Drama, School, Shounen\r\n', 'Movie\r\n', '1', '102733'), ('918', 'Gintama\r\n', 'Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen\r\n', 'TV\r\n', '201', '336376'), ('2904', 'Code Geass: Hangyaku no Lelouch R2\r\n', 'Action, Drama, Mecha, Military, Sci-Fi, Super Power\r\n', 'TV\r\n', '25', '572888'), ('28891', 'Haikyuu!! Second Season\r\n', 'Comedy, Drama, School, Shounen, Sports\r\n', 'TV\r\n', '25', '179342'), ('199', 'Sen to Chihiro no Kamikakushi\r\n', 'Adventure, Drama, Supernatural\r\n', 'Movie\r\n', '1', '466254'), ('23273', 'Shigatsu wa Kimi no Uso\r\n', 'Drama, Music, Romance, School, Shounen\r\n', 'TV\r\n', '22', '416397'), ('24701', 'Mushishi Zoku Shou 2nd Season\r\n', 'Adventure, Fantasy, Historical, Mystery, Seinen, Slice of Life, Supernatural\r\n', 'TV\r\n', '10', '75894'), ('12355', 'Ookami Kodomo no Ame to Yuki\r\n', 'Fantasy, Slice of Life\r\n', 'Movie\r\n', '1', '226193'), ('1575', 'Code Geass: Hangyaku no Lelouch\r\n', 'Action, Mecha, Military, School, Sci-Fi, Super Power\r\n', 'TV\r\n', '25', '715151'), ('263', 'Hajime no Ippo\r\n', 'Comedy, Drama, Shounen, Sports\r\n', 'TV\r\n', '75', '157670'), ('44', 'Rurouni Kenshin: Meiji Kenkaku Romantan - Tsuioku-hen\r\n', 'Action, Drama, Historical, Martial Arts, Romance, Samurai\r\n', 'OVA\r\n', '4', '129307'), ('1', 'Cowboy Bebop\r\n', 'Action, Adventure, Comedy, Drama, Sci-Fi, Space\r\n', 'TV\r\n', '26', '486824'), ('30276', 'One Punch Man\r\n', 'Action, Comedy, Parody, Sci-Fi, Seinen, Super Power, Supernatural\r\n', 'TV\r\n', '12', '552458'), ('164', 'Mononoke Hime\r\n', 'Action, Adventure, Fantasy\r\n', 'Movie\r\n', '1', '339556'), ('7311', 'Suzumiya Haruhi no Shoushitsu\r\n', 'Comedy, Mystery, Romance, School, Sci-Fi, Supernatural\r\n', 'Movie\r\n', '1', '240297'), ('17074', 'Monogatari Series: Second Season\r\n', 'Comedy, Mystery, Romance, Supernatural, Vampire\r\n', 'TV\r\n', '26', '205959'), ('21939', 'Mushishi Zoku Shou\r\n', 'Adventure, Fantasy, Historical, Mystery, Seinen, Slice of Life, Supernatural\r\n', 'TV\r\n', '10', '101351'), ('457', 'Mushishi\r\n', 'Adventure, Fantasy, Historical, Mystery, Seinen, Slice of Life, Supernatural\r\n', 'TV\r\n', '26', '300030'), ('2001', 'Tengen Toppa Gurren Lagann\r\n', 'Action, Adventure, Comedy, Mecha, Sci-Fi\r\n', 'TV\r\n', '27', '562962'), ('245', 'Great Teacher Onizuka\r\n', 'Comedy, Drama, School, Shounen, Slice of Life\r\n', 'TV\r\n', '43', '268487'), ('32983', 'Natsume Yuujinchou Go\r\n', 'Drama, Fantasy, Shoujo, Slice of Life, Supernatural\r\n', 'TV\r\n', '13', '38865'), ('5258', 'Hajime no Ippo: New Challenger\r\n', 'Comedy, Drama, Shounen, Sports\r\n', 'TV\r\n', '26', '88995'), ('28957', 'Mushishi Zoku Shou: Suzu no Shizuku\r\n', 'Adventure, Fantasy, Historical, Mystery, Seinen, Slice of Life, Supernatural\r\n', 'Movie\r\n', '1', '32266'), ('11665', 'Natsume Yuujinchou Shi\r\n', 'Drama, Fantasy, Shoujo, Slice of Life, Supernatural\r\n', 'TV\r\n', '13', '98431'), ('431', 'Howl no Ugoku Shiro\r\n', 'Adventure, Drama, Fantasy, Romance\r\n', 'Movie\r\n', '1', '333186'), ('11741', 'Fate/Zero 2nd Season\r\n', 'Action, Fantasy, Supernatural, Thriller\r\n', 'TV\r\n', '12', '340973'), ('31757', 'Kizumonogatari II: Nekketsu-hen\r\n', 'Action, Mystery, Supernatural, Vampire\r\n', 'Movie\r\n', '1', '34347'), ('19', 'Monster\r\n', 'Drama, Horror, Mystery, Police, Psychological, Seinen, Thriller\r\n', 'TV\r\n', '74', '247562'), ('12365', 'Bakuman. 3rd Season\r\n', 'Comedy, Drama, Romance, Shounen\r\n', 'TV\r\n', '25', '133620'), ('1535', 'Death Note\r\n', 'Mystery, Police, Psychological, Supernatural, Thriller\r\n', 'TV\r\n', '37', '1013917'), ('32366', 'GintamaÂ°: Aizome Kaori-hen\r\n', 'Comedy, Parody\r\n', 'OVA\r\n', '2', '16947'), ('30654', 'Ansatsu Kyoushitsu (TV) 2nd Season\r\n', 'Action, Comedy, School, Shounen\r\n', 'TV\r\n', '25', '176475'), ('20583', 'Haikyuu!!\r\n', 'Comedy, Drama, School, Shounen, Sports\r\n', 'TV\r\n', '25', '284498'), ('19647', 'Hajime no Ippo: Rising\r\n', 'Comedy, Drama, Shounen, Sports\r\n', 'TV\r\n', '25', '66756'), ('4282', 'Kara no Kyoukai 5: Mujun Rasen\r\n', 'Action, Drama, Mystery, Romance, Supernatural, Thriller\r\n', 'Movie\r\n', '1', '111074'), ('10379', 'Natsume Yuujinchou San\r\n', 'Drama, Fantasy, Shoujo, Slice of Life, Supernatural\r\n', 'TV\r\n', '13', '102322'), ('22135', 'Ping Pong The Animation\r\n', 'Psychological, Seinen, Sports\r\n', 'TV\r\n', '11', '97187'), ('21329', 'Mushishi Special: Hihamukage\r\n', 'Adventure, Fantasy, Historical, Mystery, Seinen, Slice of Life, Supernatural\r\n', 'Special\r\n', '1', '49036');

# Conclusion

This database solution works well with this data set due to the fact there isn’t many primary keys that can be made from the data set, only two are needed. Advantages of second normal form with this data set would be that all the redundant data such as genre, episodes etc are reduced effectively. Secondly, the organization of the database is improved using 2NF because of the fact there are only two primary keys in which the information only needs to be added once and the only thing in the database that has to be changed more than once would be rating (rarely though because it would only be changed if the user changes their mind on the rating they have given).

A negative of using 2NF would be that the storage later on could be an issue as there is no limit to the number of users that can rate specific anime so it can fill up storage devices. An issue with normalization in general is that it requires more tables to join. Spreading out information through multiple tables increases the need to join them.

This database is used for people who watch anime and want to give anime they have watched a rating so other people can have an idea whether the shows are worth watching. It is similar to the imdb website on how they have ratings for different shows, movies, and anime. However, mine focuses on the anime community giving ratings on anime for more valid ratings. There is a total of 3 tables in my database. One of the tables is for whenever a new user creates an account all their data will be stored in that table to prevent his/her data repeating in the main table. The other table is used to add new anime not already in the table to it with all the main information to do with it. This is useful as it will waste a lot of space repeating the same information of an anime if many users rate the same anime.

In conclusion I think the database solution I chose of second normal form was the best choice out of 1st,2nd, and 3rd normal forms as there are not many extra tables that need to be made so the number of links and foreign keys is very minimal. Due to my scenario being very fixed (something where its structure will not change) the layout and number of tables will never have to be changed. However, the only negative that I can see would be that new anime come out very often so the database table containing anime would have to be updated often. In order to fix this, I need to create a script that scrapes information from websites such as ‘MyAnimeList’ so the database can automatically update.

# References

K. Kumar and S. K. Azad, "Database normalization design pattern," 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON), 2017, pp. 318-322, doi: 10.1109/UPCON.2017.8251067. <https://ieeexplore.ieee.org/document/8251067>