## Introduction

For our database, we decided to create an anime information and watchlist database. Essentially, our database would catalog various series that fall under the anime umbrella and document facts about them such as creators, directors, studios, and so on. Our motivation for creating an anime database comes from the three of us having a passionate interest in the medium, and being deeply involved in the various communities within it. We also know finding a good database to track anime, in general, is hard to find and we would like to try our hand at it. Focusing our project around it helps us due to our familiarity with the subject, and gives us a springboard on how to develop our database and how to set up said entries and relationships. One of the main ways our database can be used is first and foremost as a source of information on the various series' information, staff, studios, air dates, and other related facts as a hub for information. Another way is that it keeps track of a studio's works and their contributions to the industry. It also allows users to track the anime they have seen or plan to see and allows them to add their favorite anime to another list. One of the important aspects of our design is that "Creator" and "Director" are separate since many series are adapted from manga series, manga being the Japanese equivalent of a graphic novel, and as such have an original Creator which is different from Director. It allows the original author to still get the credit for creating their series but also lets the director who brought the series to an animated medium get their credit as well. Another aspect is the addition of the "Manga" entity since it also allows users to have a Manga database so that they can relate different anime to their respective manga and see if any other anime was created based on the manga. There are different views when it comes to the watchlist since it will have a view of the anime the user has watched, the anime they will watch, and their favorite anime. They will also be able to give recommendations and get recommended anime based on their favorites list. Emioluwa came up with the topic of the database and all members came up with the database uses and the important aspects of it.

## Requirement Analysis

### **User Requirements**

For our database, we want to give the user specialized functionality to allow them to keep track of their favorite shows as effectively as possible.

When searching for shows, the user will be able to search by the title of the anime as well as other criteria, such as

- SEASON in which the show aired
- ANIMATION STUDIO that animated the show
- CHARACTERS that appear in the show
- STATUS of the anime (completed, ongoing)
- PLATFORM where anime is on
- Alphabetical sort of anime

The user wishes to have a database that stores anime by its generated ID number, title, number of episodes, platform, genre, type, status, release date, etc. Every anime is animated by a studio that stores the unique studio name and director’s name, but the director's name can be left empty. If there is a manga adapted into the anime then link them together and also show if any other anime have been based on the same manga. Every manga added must have a unique ID, title, release year, and genre. Every anime and manga is created by a creator. A creator can create more than one anime or manga and has to have a name and a unique ID number. At least one character is in an anime and their name, age, gender, and unique ID number are stored along with voice actor information. At least one soundtrack scores an anime. Each soundtrack has a title, artist, and unique ID. Anime shows on platforms and each platform has a unique URL and a name. Each character is voiced by a voice actor and their name along with a unique ID number is stored. Each user who makes an account must have a unique name stored. Each user can rate any anime they’ve watched and mark which are their favorites. Every user has a watchlist whether it is empty or not and has a unique ID number. Each anime on the watch list has 5 statuses to be picked from which are “Finished”, “Watching”, “Plan to Watch”, “Dropped”, or “On Hold”. Each anime on the watchlist is sorted by the anime’s name, the status of the anime, the watch status,  the platform it's on, and the rating of the anime which can be left empty.

**Specific Constraints**

- *Every* ANIME is *shown on* a PLATFORM; *Not every* PLATFORM must *show* an ANIME
- *Every* ANIME is *animated by* a STUDIO; *Not every* STUDIO must *animate* an ANIME
- *Not Every* ANIME must *be adapted* from a MANGA; *Not every* MANGA must be *adapted from* an ANIME
- *Every* ANIME is *created by* a CREATOR; *Every* CREATOR *creates* an ANIME
- *Every* ANIME *has* a CHARACTER;  *Every* CHARACTER must *appear in* an ANIME
- *Every* ANIME is *scored by* a SOUNDTRACK; *Every* SOUNDTRACK *scores* an ANIME
- *Not every* ANIME is *watched* by a USER; *Not every* USER watches ANIME
- *Every* MANGA is *created by* a CREATOR; *Not every* CREATOR *creates* MANGA
- *Every* CHARACTER is *voiced by* a VOICE_ACTOR; *Every* VOICE_ACTOR *voices* a CHARACTER
- *Every* USER *creates* one WATCHLIST; *Every* WATCHLIST is *created by* one USER

**Functional Requirement**

- The user will be able to search for shows in the database.
- The user will be able to add and remove shows to and from their watchlist.
- The user will be able to add entries to the database.
- The user will be able to edit any incorrect entries.
- The user will be able to give ratings to anime.

- The user will be required to have an account to use the site and input their rankings.
- The user's watchlist is connected directly to the user's account. If the user's account is removed, the user's watchlist will be removed as well.
- The user will be able to view other users' watchlists.
- The user will be able to search for other users.
- The user will be able to filter/sort through anime by title, genre, etc.

In this section, all members came together to figure out the database requirements including the constraints and functional requirements.

## ER Model

Since this database is for the user to add anime they have watched or plan to watch to a watchlist, we decided on a total of 11 entities being used in the anime database. They are *Anime*, *Studio*, *Manga*, *Creator*, *Character*, *Soundtrack*, *Platform*, *Voice Actor*, *User*, *Watchlist*, and *Rating*.

- *Anime*
    - takes in information about each respective anime
    - Attributes:
        - <u>animeID</u>, which is part of the composite primary key
        - <u>animeTitle,</u> which is part of the composite primary key
        - num_ep
        - anime_genre
        - type
        - status
        - rel_date
        - *studio_name, which is a foreign key that pulls from Studio*
        - *creator_ID, which is a foreign key that pulls from Creator*
        - *platform, which is a foreign key that pulls from Platform*
        - *soundtrack, which is a foreign key that pulls from Soundtrack*
        - *manga_ID, which is a foreign key that pulls from Manga*
- *Studio*
    - gives information about the animation studio that made the anime
    - Attributes:
        - <u>studioName</u>, which is the primary key attribute
        - studio_director, which takes in a director name attribute
- *Manga*
    - when an anime was created based on a manga, which is the case for most anime, and allows the user to see what manga the anime was based on
    - Attributes:
        - <u>mangaID</u>, which is the primary key attribute
        - mangaTitle

        - m_rel_date
        - manga_genre
- *Creator*
    - has the value of the creator's name and is also connected to both the *Anime* and *Manga* entities
    - Attributes:
        - creatorID, which is the key attribute
        - creatorName, which takes in the name attribute.
- *Character*
    - takes in information about characters from an anime
    - Attributes:
        - charID, which is part of the composite primary key
        - charName, which is part of the composite primary key
        - age
        - gender
        - *vaID, which pulls from Voice Actor*
        - *vaName, which pulls from Voice Actor*
- *Soundtrack*
    - takes in information about the music soundtracks that go within an anime
    - Attributes:
        - soundID, which is the primary key attribute
        - soundtrack_name
        - sound_director
- *Platform*
    - takes in information about what platforms the anime appears on
    - Attributes:
        - url, which is the primary key attribute
        - platformName
- *Voice Actor*
    - takes in information about who voice acted as a character
    - Attributes:
        - vaID, which is part of the composite primary key
        - vaName, which is part of the composite primary key
- *User*
    - for users to be able to sign in and make their watchlist and/or recommendations
    - Attribute:
        - user_ID, which is part of the composite primary key
        - userName, which is part of the composite primary key
- *Watchlist* (weak)
    - allows users to keep track of what shows they're watching

        - Attributes:
            - *userID*, to state which user
            - *userName*, to state which user
            - *animeID*, to state which anime is on the list
            - *animeTitle*, to state which anime is on the list
            - watch_status, to state the current watching status of the user
    - *Rating* (weak)
        - gives users the ability to add ratings to anime they have watched
        - Attributes:
            - *userID,* to state which user's list
            - *userName,* to state which user's list
            - *animeID*, to state which anime has been rated
            - *animeTitle*, to state which anime has been rated
            - rating
            - favorite

In this section, all 3 members worked on figuring out the entities and attributes and how to connect them through their relationships. Partial ER models were created by Emioluwa and Preston. John made the final ER model with input from the other two members.

## Relational Model

**User** (<u>userID</u>, UserName)

- In 3NF due to userID being the only primary key and having no multiple attributes. No parietal nor transitive dependencies either.

**Anime** (<u>animeID</u>, <u>animeTitle</u>, num_ep, anime_genre, type, status, rel_date*, studio_name, creator_ID, platform, soundtrack, manga_ID*)

- In 3NF due to animeID & animeTitle being the two primary keys and having no multiple attributes. No partial nor transitive dependencies either as each relation relies on the two primary keys. Generalization occurred with release date, removing our prior Season entity and combining it with the Anine entity as info on a show. Cardinality was One to Many in relation to the other tables due to the multiple shows in the database and one to one with the Manga entity.

    **Rating** (*userID*, *animeID, userName, animeTitle*, rating, favorite)

    **Watchlist** (*userID*, *userName, animeID*, *animeTitle,* watch_status)

    - Both weak entities are in 3NF, with no multiple attributes in any. No Partial or Transitive dependencies either. No generalization is needed. Both had cardinality One to Many with the Anime entity but Rating had One to One with the User Table since each rating was unique to the specific user.

**Manga** (mangaID, mangaTitle, m_rel_date, manga_genre)

- In 3NF due to mangaID being the only primary key and having no multiple attributes. No partial nor transitive dependencies either. Generalization occurred with release date as with the Anime entity to just have release date be its own attribute within the entity. Cardinality was one to one (0.1) with the Anime entity for shows that were adaptations and one to many with the Creator entity.

**Platform** (url, platformName)

- In 3NF due to url being the only primary key and having no multiple attributes. No partial nor transitive dependencies either. No generalization needed. Cardinality was one to many with the Anime entity as from Platform to Anime there had to be at least one platform for watching, but inversely not all anime were on a platform.

**Studio** (studioName, studio_director)

- In 3NF due to name being the only primary key and having no multiple attributes. No partial nor transitive dependencies either. Director was initially its own table, but it was found easier to generalize it into Studio. Cardinality was one to many both ways as each anime needed a studio to adapt it.

**Creator** (creatorID, creatorName)

- In 3NF due to creatorID being the only primary key and having no multiple attributes. No partial nor transitive dependencies either. No generalization needed. On both the Anime and Manga entities, Cardinality was one to many as each entry has at minimum one creator.

**Character** (charID, charName, age, gender, *va_ID, va_Name*)

- In 3NF due to ID & name being the two primary keys and have no multiple attributes. No partial nor transitive dependencies either as each relation relies on the two primary keys. No generalization is needed. Cardinality was one to many with Anime as each anime has a type of character to list and was zero to many with Voice Actor as not every character is voiced.

**Voice Actor** (vaID, vaName)

- In 3NF due to vaID & vaName being the two primary keys and have no multiple attributes. No partial nor transitive dependencies either as the two primary keys are the only attributes. No generalization needed. Cardinality was one to many with Character as each Voice Actor has a character, but inversely not every Character has a voice actor.

**Soundtrack** (soundID, sound_director, soundtrack_name)

- In 3NF due to ID being the only primary key and having no multiple attributes. No partial nor transitive dependencies either. No generalization is needed. Cardinality was one to many as each anime has a soundtrack and the only soundtracks included are those that appear in an anime.

In this section, all 3 members worked on figuring out how to transform the entities and relationships into relations. All 3 members also worked together to prove each relation is in 3NF. John created the final Relational Model.

## Data Dictionary

**Anime Relation**

| Field Name | Display Name | Data Type | Constraints | Character Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|
| anime_genre | Anime Genre | varchar | | 255 | Stores the genre of each anime | Y | Romance |
| num_ep | Episode Number | int | | 255 | Stores the number of episodes within an anime | Y | 24 |
| status | Status | varchar | Can only be set as “Currently Airing”, “Finished Airing”, or “Not Yet Aired” | 255 | Stores status of anime | Y | Ongoing |
| animeTitle | Anime Title | varchar | Primary | 255 | Stores each anime’s title whether it is in Japanese, English, or both | N | Fairy Tail |
| type | Anime Type | varchar | | 255 | Stores the type of each anime | Y | OVA |
| animeID | Anime ID | int | Primary; | 255 | Gives each | N | #78431 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | auto increments | | anime a unique ID number | | |
| rel_date | Release Date | year | Must be in form Year | 255 | Stores anime release date | Y | 2021 |
| studio_name | Studio Name | varchar | Foreign key referencing studio_name from Studio Relation | 255 | Stores the studio name where the anime was released; links studio information to anime | N | MAPPA Studio |
| creatorID | Creator ID | varchar | Foreign key referencing creatorID from Creator Relation | 255 | Gives each creator a unique ID number; links creator information to anime | N | #98347 |
| platform | URL | varchar | Foreign key referencing platform from Platform Relation | 255 | Stores URL of platform website; links available platform to anime | N | https://www.funimation.com/ |
| soundtrack | Soundtrack ID | int | Foreign key referencing soundtrack from Soundtrack Relation | 255 | Gives each soundtrack a unique ID number; links soundtrack information to anime | N | #340 |
| mangaID | Manga ID | int | Foreign key referencing mangaID from Manga Relation | 255 | Gives each manga a unique ID number | N | #394 |

**Studio Relation**

| Field Name | Display Name | Data Type | Constraints | Character Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|
| studioName | Studio Name | varchar | Primary, unique | 255 | Stores the studio name where the anime was released | N | MAPPA Studio |
| studio_director | Studio Director | varchar |  | 255 | Stores director's name from studio | Y | Hayao Miyazaki |

**Manga Relation**

| Field Name | Display Name | Data Type | Constraints | Character Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|
| manga_genre | Manga Genre | varchar |  | 255 | Stores the genre of each manga | Y | Action |
| mangaID | Manga ID | int | Primary; auto increments | 255 | Gives each manga a unique ID number | N | #394 |
| mangaTitle | Manga Title | varchar |  | 255 | Stores each manga its title whether it is in Japanese, English, or both | N | Fairy Tail |
| m_rel_date | Release Date | year | Must be in form Year | 10 | Stores release date of a manga | Y | 2019 |

**Creator Relation**

| Field Name | Display Name | Data Type | Constraints | Character Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|
| creatorID | Creator ID | varchar | Primary; auto increments | 255 | Gives each creator a unique ID number | N | #98347 |
| creatorName | Creator | varchar |  | 255 | Stores the creator name of anime/manga | Y | Hiro Mashisma |

**Character Relation**

| Field Name | Display Name | Data Type | Constraints | Character Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|
| age | Age | int |  | 255 | Stores a character's age | Y | 16 |
| gender | Gender | varchar |  | 255 | Stores a character's gender | Y | Male |
| charID | Character ID | int | Primary; auto increments | 255 | Gives each character a unique ID number | N | #420 |
| charName | Character Name | varchar | Primary; In the format of Last Name, First Name | 255 | Stores a character's name | N | Bakugou Katsuki |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| va_ID | Voice Actor ID | int | Foreign key referencing vaID from Voice Actor Relation | 255 | Gives each voice actor a unique ID number; links voice actor to the character being voiced | Y | #4229 |
| va_Name | Voice Actor Name | varchar | Foreign key referencing vaName from Voice Actor relation | 255 | Gives each voice actor a name; links voice actor to the character being voiced | Y | Nobuhiko Okamoto |

**Soundtrack Relation**

| Field Name | Display Name | Data Type | Constraints | Characte r Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|
| sound_director | Soundtrack Director | varchar | | 255 | Stores director name of anime soundtrack | Y | AmaLee |
| soundID | Soundtrack ID | int | Primary; auto increments | | Gives each soundtrack a unique ID number | N | #340 |
| soundtrack_name | Soundtrack Name | varchar | | 255 | Stores soundtrack name | Y | Snow Fairy |

**Platform**

| Field Name | Display Name | Data Type | Constraints | Character Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| url | URL | varchar | Primary | 255 | Stores URL of platform website | N | https://www.funimation.com/ |
| platformName | Platform | varchar | | 255 | Stores platforms that an anime is available on | N | Funimation |

**Voice Actor**

| Field Name | Display Name | Data Type | Constraints | Character Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|
| vaID | Voice Actor ID | int | Primary; auto increments | 255 | Gives each voice actor a unique ID number | N | #4229 |
| vaName | Voice Actor Name | varchar | Primary | 255 | Stores a voice actor's name | N | Nobuhiko Okamoto |

**User Relation**

| Field Name | Display Name | Data Type | Constraints | Character Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|
| userID | User ID | int | Primary, unique, auto increments | 255 | Gives each user a unique ID number | N | #08742 |
| userName | Username | varchar | Primary, unique | 25 | Stores username of each user | N | diakat |

**Watchlist Relation**

| **Field Name** | **Display Name** | **Data Type** | **Constraints** | **Character Length** | **Description** | **Accepts Null Values?** | **Acceptable Values** |
|---|---|---|---|---|---|---|---|
| animeTitle | Anime Title | int | | 255 | Gives each anime a Title; connects it to a user's watchlist | Y | #929 |
| watch_status | Status | varchar | Can only be set as "Finished", "Watching", "Plan to Watch", "Dropped", or "On Hold" | 255 | Allows users to pick the status of the anime they are watching | Y | Finished |
| userID | User ID | int | Foreign key referencing userID from User Relation | 255 | Gives each user a unique ID number; links user information to watchlist system | N | #08742 |
| userName | Username | varchar | Foreign key referencing userName from User relation | 25 | Gives each user a name; links info to Watchlist | N | diakat |
| animeID | Anime ID | int | Foreign key referencing aniID from Anime Relation | 255 | Gives each anime a unique ID number; links anime to watchlist system | Y | #1027 |

**Rating Relation**

| Field Name | Display Name | Data Type | Constraints | Character Length | Description | Accepts Null Values? | Acceptable Values |
|---|---|---|---|---|---|---|---|
| rating | Rating | int | Stores integers from 0 to 10 | 2 | Allows user to rate anime watched | Y | 9 |
| userID | User ID | int | Foreign key referencing userID from User Relation | 255 | Gives each user a unique ID number; links user to the rating system | Y | #08742 |
| userName | Username | varchar | Foreign key referencing userName from User relation | 25 | Gives each user a name, links it to the rating system | N | diakat |
| animeID | Anime ID | int | Foreign key referencing animeID from Anime Relation | 255 | Gives each anime a unique ID number; links anime to the rating system | Y | #3072 |
| animeTitle | Anime Title | varchar | Foreign key referencing animeTitle | 255 | Gives each anime a Title; links anime to the rating system | Y | Boku No Hero Academia |
| favorite | Favorite | boolean | Automatically set as false | 1 | Lets user select which anime is their favorite (true or false) | Y | True |

Emioluwa worked on the data dictionary with input from the other team members for agreement on the field and display names, constraints, etc.

## Implementation

The implementation of our code to ensure all the user and functional requirements were met. Some of those requirements were to make sure users will have the ability to search for and filter through shows without hassle. They would be able to add and remove shows from their watchlists. They would be able to add anime directly to the database if it is not already there as well as edit any incorrect entries. It was made sure that each requirement has a matching query so it is shown through the database. Some of the queries used are

```
SELECT *

FROM animedb.Anime A

ORDER BY A.type;
```

which is used for sorting/filtering anime through the database or if an user 'joenuts' wanted to add the "Hunter x Hunter (2011)" into their watchlist then the query

```
INSERT INTO animedb.Watchlist(userName, animeTitle, watch_status)

VALUES ('joenuts','Hunter x Hunter (2011)', 'Watching');
```

would be used to perform that action. This section was worked on by Preston who mainly did the SQL code with input from the other team members to ensure all the names, constraints, relations, etc. matched to the data dictionary, relational model, and the ER model.

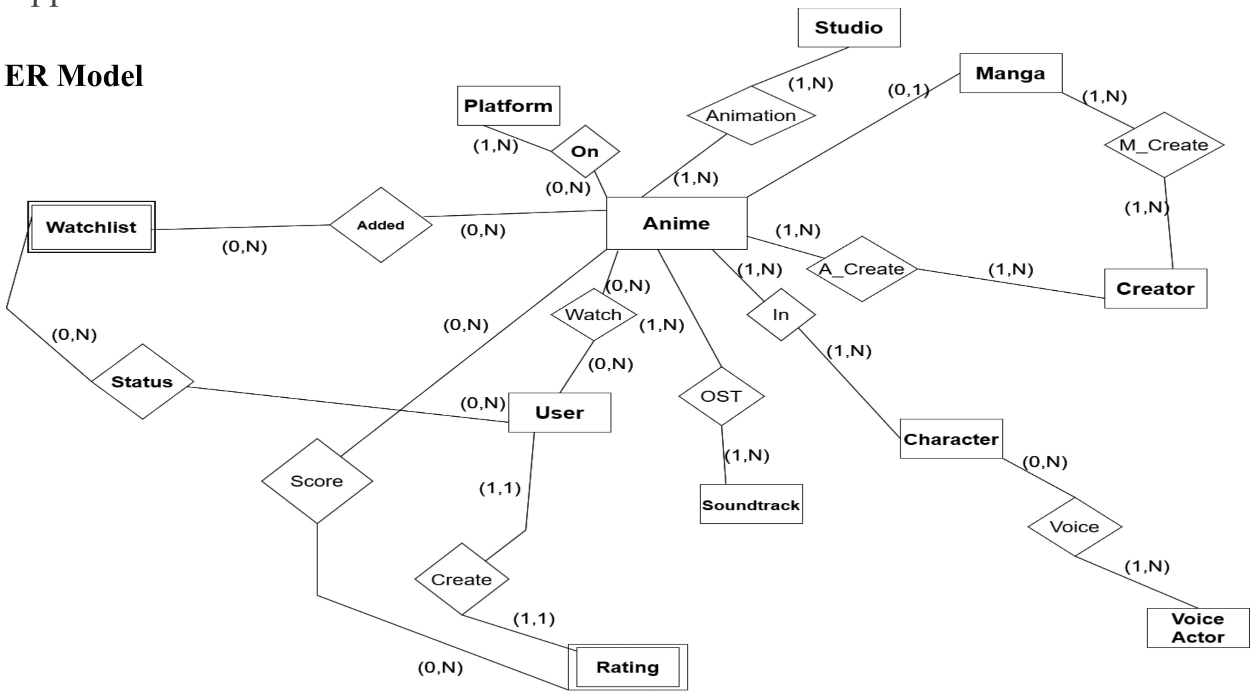
## Summary

The anime database took a lot of trial and error to ensure we met all the user and functional requirements. We had to remove some relations and replace them with new ones. We had to get rid of some ideas to make the database easier to use and query. We wanted to make sure the anime database would be easy to use for the users. Also, we wanted the users to be able to interact with other users as well as have as much information on each anime added to the database. The project idea came from Emioluwa and was agreed upon by all members. All members came together to figure out the uses for the database as well as the important aspects we wanted to focus on. The Introduction section was written by all the members. The Requirement Analysis section was broken up into parts. Emioluwa and Preston wrote the user requirements. John wrote the specific constraints we wanted to focus on. All the members had input into the functional requirements for the database. The final ER Model was done by John while Emioluwa and Preston worked on what entities and attributes would be needed for the ER Model. The Relational Model section was mainly done by John who completed the final Relational Model and explained why each relation was in 3NF. The Data Dictionary section was done by Emioluwa with input from the other members on the constraints and such. The Implementation section was done by Preston with help from the other members to ensure the queries and everything worked correctly. The Demo Video also was done by Preston.

**Demo Video**

## Appendix

### ER Model



### Relational Model

**User** (userID, UserName)

**Anime** (animeID, animeTitle, num_ep, genre, type, status, rel_date, *studio_name*, *creatorID*, *platform*, *soundtrack*, *mangaID*)

**Rating** (*userID*, *animeID*, *userName*, *animeTitle*, rating, favorite)

**Watchlist** (*userID*, *userName*, *animeID*, *animeTitle*, watch_status)

**Manga** (mangaID, mangaTitle, rel_date, genre)

**Platform** (url, name)

**Studio** (name, director)

**Creator** (creatorID, creatorName)

**Character** (ID, name, age, gender, *vaID*, *vaName*)

**Voice Actor** (vaID, vaName)

**Soundtrack** (ID, artist, name)

SQL Folder