Basic Database Final project report

Database for Anime website

Group 1 – BI10

University of Science and Technology of Hanoi

December 2020

Table of Contents

1. Introduction

1.1. Overview

Anime is hand-drawn and computer-animation originating and produced in Japan with its traditional, typical style. Nowadays, anime is extremely trending and famous, especially among teenagers, because of the variety of amazing plots and deeply conveyed morals. Therefore, anime websites were created so that people can manage to watch animes more conveniently.

In this report, we will develop a simple design for a database of an anime website. First, we will specify the user requirements, then sketch an entity relationship diagram, represent the schema, and finally we will implement some simple SQL queries to deploy the functionalities.

1.2. Group members

Dương Đăng Hưng	BI10-073
Tạ Quang Hiếu	BI10-065
Đỗ Quang Hiếu	BI10-063
Nguyễn Hoàng Minh	BI10-112
Đỗ Hoàng Quân	BI10-147

2. User requirements

For an anime website, we mainly focus on two types of users: the website managers (or administrators) and the watchers (or viewers). Each types may be able to use different functionalities, since they have different purposes.

2.1. Website managers

People who control the website and in charge of managing the database should be able to:

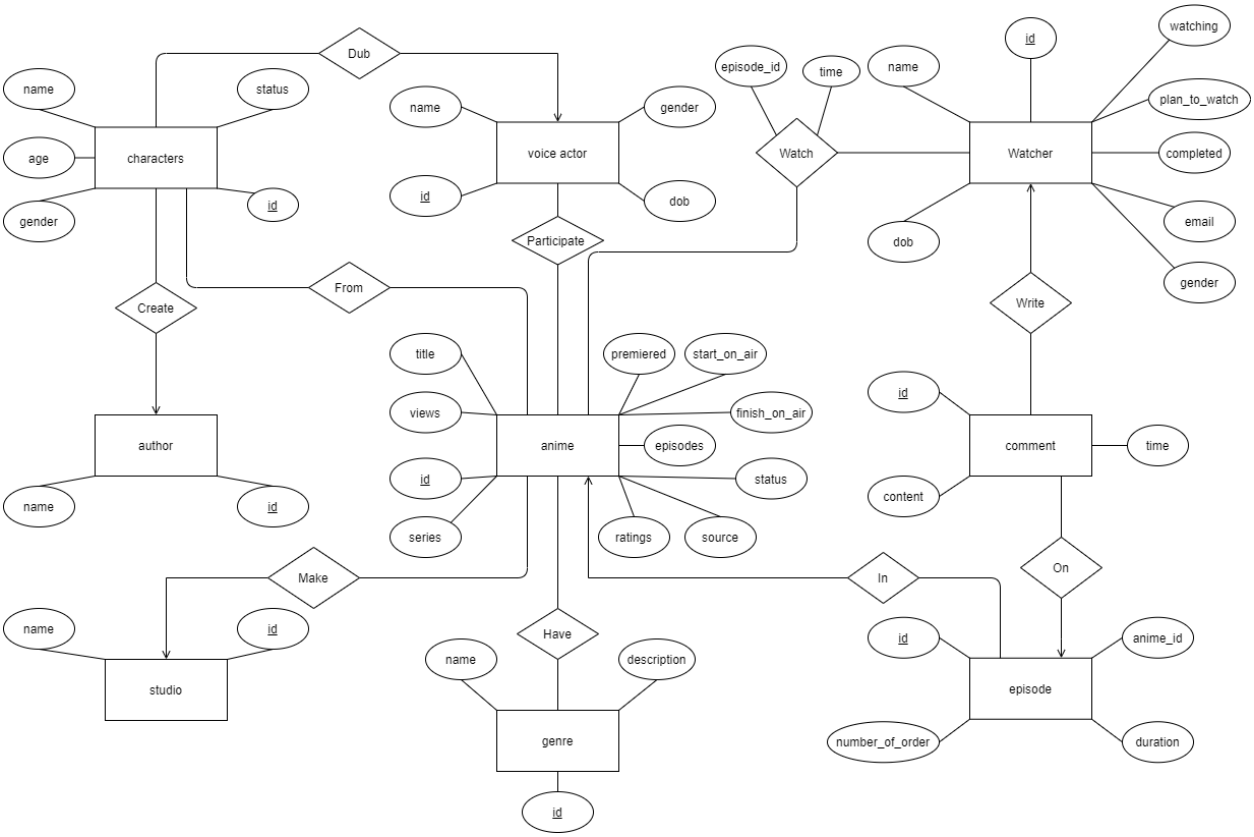
- Manage an anime's information: upload, view, edit, delete.
- Upload an anime's new episodes, as well as delete existed episodes.
- Create, view, update some lists such as "Top trending anime list".
- Check the number of views of an anime.
- Manage watchers' comments: view, edit, delete.

2.2. Watchers

Watchers should be provided with most simple and convenient functionalities such as:

- Manage his/her account, including:
    - Register a new account
    - Edit information: name, gender, email address, date of birth.
    - Record watch history
    - Delete account
- Write, edit, delete comments.
- Check the number of views of an anime.
- Mark an anime as favorite.
- Mark an anime as plan-to-watch.
- View self-account: history, favorite, watching, plan-to-watch, completed.
- View other watchers' accounts.
- Use the search filter to:
    - Find an anime based on its title, character
    - Filter animes with specific genres, status, author, etc.
    - Sort animes by views.

3. Entity-Relationship Diagram



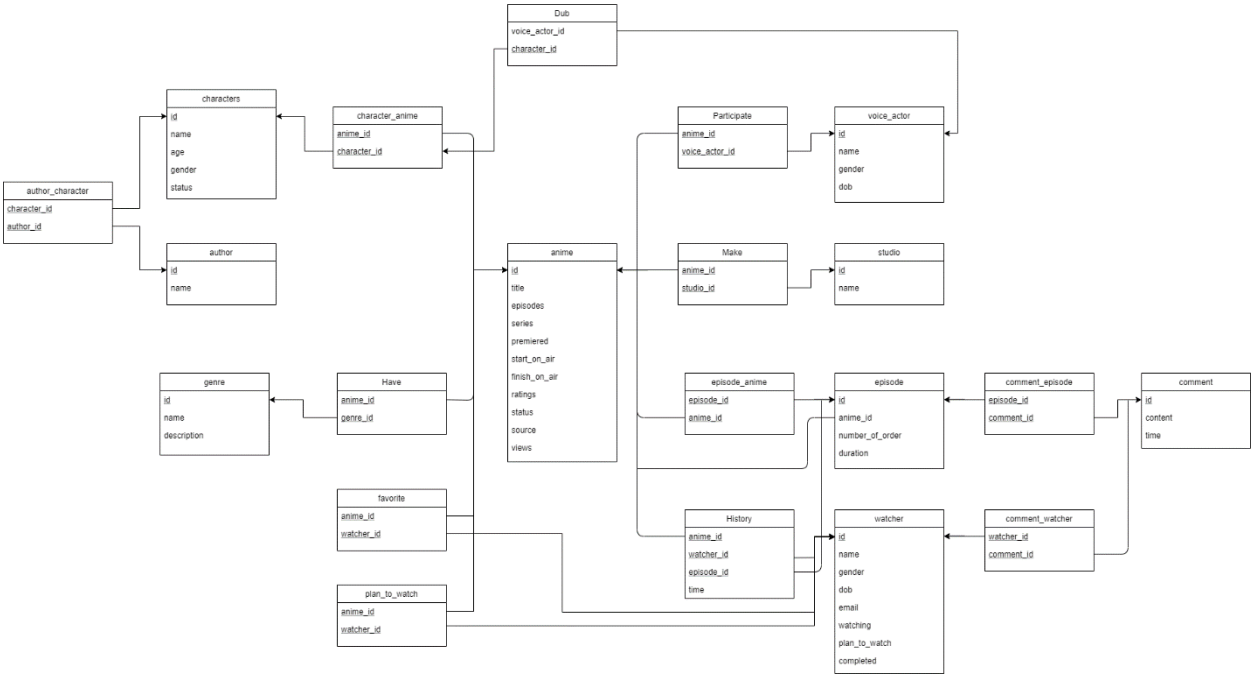
Further explanations on relationships:

- anime – genre (Have): many – many, one anime can have many genres, and one genre can also belongs to many animes.
- anime – studio (Make): many – one, one studio can make many animes, but one anime can only be made by one studio.
- anime – watcher (Watch): many – many, one anime can be watched by many watchers, and one watcher can watch many animes.
- characters – anime (From): many – many, one anime can have many characters, and one character can also appear in many seasons (which are considered different animes) throughout a series.
- characters – author (Create): Many – one, one author can create many characters, but one character can only be created by one author.
- characters – voice_actor (Dub): Many – one, one voice actor can dub for many characters, but one character can only be dubbed by one voice actor.
- voice_actor – anime (Participate): many – many, one voice actor can participate in many animes, and one anime can have many voice actors.
- episode – anime (In): Many – one, one anime can have many episodes, but an episode can only be in one anime.

- comment – watcher (Write): Many – one, one watcher can write many comments, but a comment can only be written by one watcher.
- comment – episode (On): Many – one, one episode can have many comments, but a comment can only be on one specific episode of one specific anime

4. Database schema

Based on the ERD and user requirements, we have developed a schema with 22 relations, which are all in 3NF.



Database schema description:

- anime (id, title, episodes, series, premiered, start_on_air, finish_on_air, ratings, status, source, average_score, views): General information of the animes.
- author (id, name): Name of the authors.
- genre (id, name, description): Name and brief description of the genres.
- episode (id, anime_id, number_of_order, duration): General information of a specific episode in an anime.
- characters (id, name, gender, age, status): General information of the characters.
- voice_actor (id, name, gender, dob): General information of the voice actors.
- studio (id, name): Name of the studios .
- watcher (id, name, gender, dob, email, watching, plan_to_watch, completed): General information of the watchers, including the number of animes that the watcher is still watching, planning to watch, and completed watching.

- comment (id, content, time): Content of the comment, identity of the watcher who wrote the comment, and the anime where the comment is written.
- favorite (watcher_id, anime_id): List of watcher's favorite animes.
- plan_to_watch (watcher_id, anime_id): List of watcher's plan-to-watch animes.
- Have (anime_id, genre_id): Represents the "Have" relationship between anime and genre.
- History (watcher_id, anime_id, episode_id, time): Represents the "Watch" relationship between episode and watcher.
- Make (studio_id, anime_id): Represents the "Make" relationship between studio and anime.
- character_anime (characters_id, anime_id): Represents the "From" relationship between character and anime.
- Dub (characters_id, voice_actor_id): Represents the "Dub" relationship between character and voice actor.
- author_character (author_id, characters_id): Represents the "Create" relationship between author and character.
- Participate (voice_actor_id, anime_id): Represents the "Participate" relationship between voice_actor and anime.
- episode_anime (episode_id, anime_id): Represents the "In" relationship between episode and anime.
- comment_watcher (watcher_id, comment_id): Represents the "Write" relationship between watcher and comment.
- comment_episode (comment_id, episode_id): Represents the "On" relationship between comment and episode.

5. Implement databases using MySQL

In this part, since we have so many relations (including entity sets and relationships among them) and stored procedures, the SQL statements would be *very long* (about more than 300 lines). So we will only mention about the most fundamental and typical ones. For more details, you can take a look at the SQL files.

5.1. Create objects statements (tables)

- anime:

```
CREATE TABLE IF NOT EXISTS anime (
        id INT NOT NULL AUTO_INCREMENT,
    title VARCHAR(100),
    episodes VARCHAR(100),
    series VARCHAR(100),
       premiered VARCHAR(100),
    start_on_air DATE,
    finish_on_air DATE,
    ratings    VARCHAR(100),
    `status` VARCHAR(100) DEFAULT 'unknown',
    `source` VARCHAR(100),
    views INT DEFAULT 0,
    author_id INT NOT NULL,
    FOREIGN KEY (author_id) REFERENCES author (id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
    UNIQUE (title),
    PRIMARY KEY (id)
);
```

- author:

```
CREATE TABLE IF NOT EXISTS author (
        id INT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(100) DEFAULT 'Unknown',
    UNIQUE (id),
    PRIMARY KEY (id)
);
```

- episode:

```
CREATE TABLE IF NOT EXISTS episode (
        id INT NOT NULL AUTO_INCREMENT,
    anime_id INT NOT NULL,
    number_of_order VARCHAR(100),
    duration VARCHAR(20) DEFAULT '24 Mins',
    PRIMARY KEY (id),
    FOREIGN KEY (anime_id) REFERENCES anime(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE );
```

- characters:

```
CREATE TABLE IF NOT EXISTS characters (
       id INT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(100),
    age VARCHAR(100),
    gender VARCHAR(20) DEFAULT 'unspecified',
    `status` VARCHAR(100),
    PRIMARY KEY (id)
);
```

- genre:

```
CREATE TABLE IF NOT EXISTS genre (
       id INT NOT NULL AUTO_INCREMENT,
       `name` VARCHAR(100),
    `description` VARCHAR(1000),
    PRIMARY KEY (id)
);
```

- voice_actor:

```
CREATE TABLE IF NOT EXISTS voice_actor (
       id INT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(100) NOT NULL,
    gender VARCHAR(20) DEFAULT 'unspecified',
    dob DATE,
    PRIMARY KEY (id)
);
```

- studio:

```
CREATE TABLE IF NOT EXISTS studio (
       id INT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(100),
    PRIMARY KEY (id)
);
```

- watcher:

```
CREATE TABLE IF NOT EXISTS watcher (
       id INT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(100),
    `gender` VARCHAR(20) DEFAULT 'unspecified',
    dob DATE,
    email VARCHAR(100) NOT NULL,
    watching INT DEFAULT 0,
    plan_to_watch INT DEFAULT 0,
    completed INT DEFAULT 0,
    UNIQUE (`name`),
    UNIQUE (email),
    PRIMARY KEY (id));
```

- comments:

```
CREATE TABLE IF NOT EXISTS comments (
       id INT NOT NULL AUTO_INCREMENT,
    content VARCHAR(1000000),
    `time` DATETIME,
    PRIMARY KEY (id)
);
```

- favorite:

```
CREATE TABLE IF NOT EXISTS favorite (
       watcher_id INT NOT NULL,
    anime_id INT NOT NULL,
    PRIMARY KEY (watcher_id, anime_id),
    FOREIGN KEY (watcher_id) REFERENCES watcher(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
    FOREIGN KEY (anime_id) REFERENCES anime(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
```

- plan_to_watch:

```
CREATE TABLE IF NOT EXISTS plan_to_watch (
       watcher_id INT NOT NULL,
    anime_id INT NOT NULL,
    PRIMARY KEY (watcher_id, anime_id),
    FOREIGN KEY (watcher_id) REFERENCES watcher(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
    FOREIGN KEY (anime_id) REFERENCES anime(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE
);
```

- History:

```
CREATE TABLE IF NOT EXISTS `History` (
       watcher_id INT NOT NULL,
    anime_id INT NOT NULL,
    episode_id INT NOT NULL,
    `time` DATE,
    PRIMARY KEY (watcher_id, anime_id, episode_id),
    FOREIGN KEY (watcher_id) REFERENCES watcher(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
    FOREIGN KEY (anime_id) REFERENCES anime(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,
    FOREIGN KEY (episode_id) REFERENCES episode(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE );
```

And the relationships… (See more in SQL files)

5.2. Insert sample rows

- Insert into “author”:

```
INSERT INTO author (name) VALUES ("Shinkai Makoto");
INSERT INTO author (name) VALUES ("Fujiko Fujio");
INSERT INTO author (name) VALUES ("Miyazaki Hayao");
```

- Insert into “anime”:

```
INSERT INTO anime (title, episodes, series, premiered, start_on_air,
finish_on_air, ratings, `status`, `source`, author_id) VALUES ("Spirited
Away", "1", "Spirited Away", "Autumn 2001", "2001-07-20","2001-07-20","PG
- Children","Finished Airing","Original", 3);
INSERT INTO anime (title, episodes, series, premiered, start_on_air,
finish_on_air, ratings, `status`, `source`, author_id) VALUES ("Howl's
Moving Castle", "1", "Howl's Moving Castle", "Winter 2004", "2004-11-
20","2004-11-20","G - All Ages","Finished Airing","Novel", 3);
INSERT INTO anime (title, episodes, series, premiered, start_on_air,
finish_on_air, ratings, `status`, `source`, author_id) VALUES ("Your
Name", "1", "Your Name", "Autumn 2016", "2016-08-26","2016-08-26","PG -
13","Finished Airing","Original", 1);
INSERT INTO anime (title, episodes, series, premiered, start_on_air,
finish_on_air, ratings, `status`, `source`, author_id) VALUES ("Princess
Mononoke", "1", "Princess Mononoke", "Autumn 1997", "1997-07-12","1997-07-
12","PG - 13","Finished Airing","Original", 3);
INSERT INTO anime (title, episodes, series, premiered, start_on_air,
finish_on_air, ratings, `status`, `source`, author_id) VALUES ("Doraemon",
"1787", "Doraemon", "Spring 1979", "1979-04-02","2005-03-18","PG -
Children","Finished Airing","Manga", 2);
INSERT INTO anime (title, episodes, series, premiered, start_on_air,
finish_on_air, ratings, `status`, `source`, author_id) VALUES ("Doraemon
the Movie: Nobita and the Windmasters", "1", "Doraemon", "Spring 2003",
"2003-03-08","2003-03-08","PG - Children","Finished Airing","Manga", 2);
```

- Insert into “episode”:

```
INSERT INTO episode (anime_id,number_of_order,duration) VALUES (1,1,"2
Hrs. 5 Mins");
INSERT INTO episode (anime_id,number_of_order,duration) VALUES (2,1,"1 Hr.
59 Mins");
INSERT INTO episode (anime_id,number_of_order,duration) VALUES (3,1,"1 Hr.
46 Mins");
INSERT INTO episode (anime_id,number_of_order,duration) VALUES (4,1,"2
Hrs. 15 Mins");
INSERT INTO episode (anime_id,number_of_order,duration) VALUES (5,1,"11
Mins");
INSERT INTO episode (anime_id,number_of_order,duration) VALUES (5,2,"11
Mins");
```

```
INSERT INTO episode (anime_id,number_of_order,duration) VALUES (5,3,"11
Mins");
INSERT INTO episode (anime_id,number_of_order,duration) VALUES (5,4,"11
Mins");
INSERT INTO episode (anime_id,number_of_order,duration) VALUES (6,1,"1 Hr.
20 Mins");
```

- Insert into “characters”:

```
INSERT INTO characters (`name`,age,gender,`status`) VALUES
("Haku",12,"Male","Alive");
INSERT INTO characters (`name`,age,gender,`status`) VALUES ("Chihiro
Ogino",12,"Female","Alive");
INSERT INTO characters (`name`,age,gender,`status`) VALUES
("Howl",27,"Male","Alive");
INSERT INTO characters (`name`,age,gender,`status`) VALUES ("Sophie
Hatter",18,"Female","Alive");
INSERT INTO characters (`name`,age,gender,`status`) VALUES ("Tachibana
Taki",23,"Male","Alive");
INSERT INTO characters (`name`,age,gender,`status`) VALUES ("Miyamizu
Mitsuha",26,"Female","Alive");
INSERT INTO characters (`name`,age,gender,`status`) VALUES
("San",16,"Female","Alive");
INSERT INTO characters (`name`,age,gender,`status`) VALUES
("Ashitaka",17,"Male","Alive");
INSERT INTO characters (`name`,age,gender,`status`) VALUES ("Nobi
Nobita",10,"Male","Alive");
INSERT INTO characters (`name`,age,`status`) VALUES
("Doraemon",10,"Alive");
```

- Insert into “genre”:

```
INSERT INTO genre (`name`, `description`) VALUES ("Fantasy","A genre of
speculative fiction set in a fictional universe, inspired by myth and
folklore");
INSERT INTO genre (`name`, `description`) VALUES ("Adventure","A genre of
film whose plots feature elements of travel");
INSERT INTO genre (`name`, `description`) VALUES ("Supernatural","A genre
of speculative fiction that exploits or is centered on supernatural
themes");
INSERT INTO genre (`name`, `description`) VALUES ("Drama","A genre of
narrative fiction intended to be serious in tone, focusing on in-depth
development of characters who must deal with emotional struggles");
INSERT INTO genre (`name`, `description`) VALUES ("Romance","Primarily
focused on the relationship between the main characters of the story");
INSERT INTO genre (`name`, `description`) VALUES ("School","Centering on
school-life");
INSERT INTO genre (`name`, `description`) VALUES ("Action","The main
character usually takes a risky turn which leads to desperate
situations");
INSERT INTO genre (`name`, `description`) VALUES ("Comedy","Tells about a
series of funny or comical events intended to make the audience laugh");
INSERT INTO genre (`name`, `description`) VALUES ("Kids","Innocent and
easy-to-understand storyline, suitable for children");
```

- Insert into “voice_actor”:

```
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Miyu Irino",
"Male", "1988-02-19");
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Rumi Hiiragi",
"Female", "1987-08-01");
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Takuya Kimura",
"Male", "1972-11-13");
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Chieko Baisho",
"Female", "1941-06-29");
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Ryuunosuke Kamiki",
"Male", "1993-05-19");
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Mone
Kamishiraishi", "Female", "1998-01-27");
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Yuriko Ishida",
"Female", "1969-10-03");
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Youji Matsuda",
"Male", "1967-10-19");
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Noriko Ohara",
"Female", "1935-10-02");
INSERT INTO voice_actor (`name`, gender, dob) VALUES ("Nobuyo Ooyama",
"Female", "1933-10-16");
```

- Insert into “studio”:

```
INSERT INTO studio (`name`) VALUES ("Ghibli");
INSERT INTO studio (`name`) VALUES ("CoMix Wave Films");
INSERT INTO studio (`name`) VALUES ("Shin-Ei Animation");
```

- Insert into “watcher”:

```
INSERT INTO watcher (`name`,gender,dob,email) VALUES
("bellkirato","Male","2001-07-22","hungdd.bi10-073@st.usth.edu.vn");
INSERT INTO watcher (`name`,gender,dob,email) VALUES
("ctn3m0","Male","2001-08-15","hieutq.bi10-065@st.usth.edu.vn");
INSERT INTO watcher (`name`,gender,dob,email) VALUES
("zer0warm","Male","1997-09-20","hieudq.bi10-063@st.usth.edu.vn");
INSERT INTO watcher (`name`,gender,dob,email) VALUES ("homi","Male","2001-
05-19","minhnh.bi10-112@st.usth.edu.vn");
INSERT INTO watcher (`name`,gender,dob,email) VALUES ("kwan","Male","2001-
11-01","quandh.bi10-147@st.usth.edu.vn");
```

- Insert into “comments”:

```
INSERT INTO comments (content,`time`) VALUES ("Very good", "2020-12-08
01:41:00");
INSERT INTO comments (content,`time`) VALUES ("Great Anime!", "2020-12-08
00:41:00");
INSERT INTO comments (content,`time`) VALUES ("I love Mitsuha", "2020-12-
08 19:21:00");
```

- Insert into “favorite”:

```
INSERT INTO favorite (watcher_id, anime_id) VALUES (1,1);
INSERT INTO favorite (watcher_id, anime_id) VALUES (1,2);
INSERT INTO favorite (watcher_id, anime_id) VALUES (1,3);
INSERT INTO favorite (watcher_id, anime_id) VALUES (2,1);
INSERT INTO favorite (watcher_id, anime_id) VALUES (2,4);
INSERT INTO favorite (watcher_id, anime_id) VALUES (3,5);
```

- Insert into “plan_to_watch”:

```
INSERT INTO plan_to_watch (watcher_id, anime_id) VALUES (1,4);
INSERT INTO plan_to_watch (watcher_id, anime_id) VALUES (1,5);
INSERT INTO plan_to_watch (watcher_id, anime_id) VALUES (2,2);
INSERT INTO plan_to_watch (watcher_id, anime_id) VALUES (2,3);
INSERT INTO plan_to_watch (watcher_id, anime_id) VALUES (3,1);
INSERT INTO plan_to_watch (watcher_id, anime_id) VALUES (3,2);
```

- Insert into “History”:

```
INSERT INTO `History` (watcher_id, anime_id, episode_id, `time`) VALUES
(1,1,1,"2020-07-22");
INSERT INTO `History` (watcher_id, anime_id, episode_id, `time`) VALUES
(1,2,2,"2020-08-20");
INSERT INTO `History` (watcher_id, anime_id, episode_id, `time`) VALUES
(2,1,1,"2019-12-24");
```

And insert sample rows into the relationships based on the data we had (See more in SQL files).

5.3. Website managers’ requirements

- Manage an anime’s information:

    + Upload a new anime:

```
DELIMITER $$
CREATE PROCEDURE upload_anime(IN title VARCHAR(100), episodes
VARCHAR(100), series VARCHAR(100), premiered VARCHAR(100), start_on_air
DATE, finish_on_air DATE, ratings VARCHAR(100), `source` VARCHAR(100),
author_id INT)
BEGIN
       INSERT INTO anime (title, episodes, series, premiered,
start_on_air, finish_on_air, ratings, `source`, views, author_id)
       VALUES (title, episodes, series, premiered, start_on_air,
finish_on_air, ratings, `source`, author_id);
END $$
DELIMITER ;
```

+ View an anime's information

```
        DELIMITER $$
CREATE PROCEDURE view_anime(IN anime_id INT)
BEGIN
        SELECT * FROM anime
    WHERE id = anime_id;
END $$
DELIMITER ;
```

+ Edit an anime's information:

```
DELIMITER $$
CREATE PROCEDURE edit_anime(IN anime_id INT, title VARCHAR(100), episodes
VARCHAR(100),series VARCHAR(100),premiered VARCHAR(100),start_on_air
DATE,finish_on_air DATE,ratings VARCHAR(100),`status`
VARCHAR(100),`source` VARCHAR(100), author_id INT)
BEGIN
        UPDATE anime
    SET title = title, episodes = episodes, series = series, premiered =
premiered, start_on_air = start_on_air, finish_on_air = finish_on_air,
ratings = ratings, `status` = `status`, `source` = `source`, author_id =
author_id
    WHERE id = anime_id;
END $$
DELIMITER ;
```

+ Delete an anime

```
DELIMITER $$
CREATE PROCEDURE delete_anime(IN anime_id INT)
BEGIN
        DELETE FROM anime WHERE id = anime_id;
END $$
DELIMITER ;
```

- Since we have `ON UPDATE CASCADE` and `ON DELETE CASCADE` in every foreign key, we do not have to edit/delete the relevant information in other relations manually, instead, it is automatically edited/deleted.

- Upload a new episode:

```
DELIMITER $$
CREATE PROCEDURE upload_episode(IN anime_id INT, number_of_order
VARCHAR(100), duration VARCHAR(20))
BEGIN
```

```
        INSERT INTO episode (anime_id, number_of_order, duration) VALUES
(anime_id, number_of_order, duration);
    INSERT INTO episode_anime (episode_id, anime_id) VALUES
(MAX(episode.id), anime_id);
END $$
DELIMITER ;
```

- Delete an anime's particular episode:

```
DELIMITER $$
CREATE PROCEDURE delete_episode(IN episode_id INT)
BEGIN
        DELETE FROM episode WHERE id = episode_id;
END $$
DELIMITER ;
```

- Create and update a "Top trending anime" list based on views:

```
DELIMITER $$
CREATE PROCEDURE update_top_trending()
BEGIN
DROP TABLE IF EXISTS trend;
CREATE TABLE trend (
        SELECT title, views
        FROM anime
    ORDER BY views DESC
    LIMIT 0,10
);
END $$
DELIMITER ;
```

- View/display "Top trending anime"

```
DELIMITER $$
CREATE PROCEDURE display_top_trending()
BEGIN
        SELECT * FROM trend;
END $$
DELIMITER ;
```

- Check the number of views of an anime:

```
DELIMITER $$
CREATE PROCEDURE check_view(IN anime_id INT)
BEGIN
        SELECT id, title, views
    FROM anime
    WHERE id = anime_id;
END $$
```

```
DELIMITER ;
```

- Manage watchers' comment:

+ View:

```
DELIMITER $$
CREATE PROCEDURE view_comment(IN comment_id INT)
BEGIN
       SELECT `time`, watcher.`name`,content
    FROM comments
               JOIN comment_watcher ON comments.id =
comment_watcher.comment_id
               JOIN watcher ON watcher.id = comment_watcher.watcher_id
    WHERE comments.id = comment_id;
END $$
DELIMITER ;
```

+ Edit:

```
DELIMITER $$
CREATE PROCEDURE admin_edit_comment(IN comment_id INT, content
VARCHAR(1000000))
BEGIN
       UPDATE comments SET content = content WHERE id = comment_id;
END $$
DELIMITER ;
```

+ Delete:

```
DELIMITER $$
CREATE PROCEDURE delete_comment(IN comment_id INT)
BEGIN
       DELETE FROM comments WHERE id = comment_id;
END $$
DELIMITER ;
```

- Beside the fundamental functions listed above, we created much more stored procedures for website managers' purposes such as: upload/edit new author, studio, genre, characters; declare that a studio/author made an anime; declare that a character belongs to an anime; etc. If you want to see more details, please check our SQL files.

5.4. Watchers' requirements

- Register a new account:

```
DELIMITER $$
CREATE PROCEDURE register_account(IN `name` VARCHAR(100), gender
VARCHAR(20), dob DATE, email VARCHAR(100))
BEGIN
	INSERT INTO watcher (`name`,gender, dob, email) VALUES
(`name`,gender, dob, email);
END $$
DELIMITER ;
```

- Edit his/her account's information:

```
DELIMITER $$
CREATE PROCEDURE update_account(IN watcher_id INT, `name` VARCHAR(100),
gender VARCHAR(20), dob DATE, email VARCHAR(100))
BEGIN
	UPDATE watcher
    SET `name` = `name`, gender=gender, dob = dob, email= email
    WHERE id = watcher_id;
END $$
DELIMITER ;
```

- Record watch history (and increase the view of the anime watched by 1):

```
DELIMITER $$
CREATE PROCEDURE record_history(IN watcher_id INT, anime_id INT,
episode_id INT, `time` DATE)
BEGIN
	INSERT INTO `History` VALUES (anime_id, watcher_id, episode_id,
`time`);
    UPDATE anime
		SET views = views + 1
        WHERE id = anime_id;
END $$
DELIMITER ;
```

- Delete his/her own account:

```
DELIMITER $$
CREATE PROCEDURE del_acc(IN watcher_id INT)
BEGIN
	DELETE FROM watcher
	WHERE id = watcher_id;
END $$
DELIMITER ;
```

- Write a comment:

```
DELIMITER $$
CREATE PROCEDURE add_comment(IN watcher_id INT, episode_id INT, content
VARCHAR(1000000),`time` DATETIME)
BEGIN
	INSERT INTO comments (content,`time`) VALUES (content, `time`);
	INSERT INTO comment_watcher (comment_id, watcher_id) VALUES
(MAX(comments.id),watcher_id);
	INSERT INTO comment_episode (comment_id, episode_id) VALUES
(MAX(comments.id),episode_id);
END $$
DELIMITER ;
```

- Edit a comment:

```
DELIMITER $$
CREATE PROCEDURE watcher_edit_comment(IN comment_id INT,content
VARCHAR(1000000))
BEGIN
	UPDATE comments
	SET content = content
	WHERE id = comment_id;
END $$
DELIMITER ;
```

- Delete a comment:

```
DELIMITER $$
CREATE PROCEDURE watcher_del_comment(IN comment_id INT)
BEGIN
	DELETE FROM comments
	WHERE id = comment_id;
END $$
DELIMITER ;
```

- View all his/her comments:

```
DELIMITER $$
CREATE PROCEDURE watcher_view_comment(IN watcher_id INT)
BEGIN
	SELECT anime.title AS `anime`, episode.number_of_order AS
`episode`, DATE_FORMAT(comments.`time`, "%a, %d %b %Y - %T") AS
`timestamp`, comments.content
	FROM comments
	INNER JOIN comment_watcher ON comments.id =
comment_watcher.comment_id
	INNER JOIN comment_episode ON comments.id =
comment_episode.comment_id
	INNER JOIN episode ON comment_episode.episode_id = episode.id
	INNER JOIN anime ON episode.anime_id = anime.id
	WHERE comment_watcher.watcher_id = watcher_id;
END $$
DELIMITER ;
```

- Check the number of views of an anime:

```
DELIMITER $$
CREATE PROCEDURE watcher_check_view(IN anime_id INT)
BEGIN
	SELECT id, title, views
    FROM anime
    WHERE id = anime_id;
END $$
DELIMITER ;
```

- Mark an anime as “plan to watch”:

```
DELIMITER $$
CREATE PROCEDURE watcher_plan(IN watcher_id INT, anime_id INT)
BEGIN
	INSERT INTO plan_to_watch (watcher_id, anime_id) VALUES
(watcher_id,anime_id);
END $$
DELIMITER ;
```

- Mark an anime as “favorite”:

```
DELIMITER $$
CREATE PROCEDURE watcher_favorite(IN watcher_id INT, anime_id INT)
BEGIN
	INSERT INTO favorite (watcher_id, anime_id) VALUES
(watcher_id,anime_id);
END $$
DELIMITER ;
```

- Use the search filter:

+ Search for an anime based on its title. For example, here, we want to search for anime whose title contains the word “Movie”:

```
SELECT * FROM anime
WHERE (title LIKE '%Movie%');
```

+ Search for an anime based on its characters. For the example below, we want to search for an anime that has the character name “San”:

```
SELECT title, episodes, series, premiered, start_on_air, finish_on_air,
ratings, anime.`status`, `source`, views, author.`name` as "Author name"
FROM character_anime
       INNER JOIN characters ON character_anime.characters_id =
characters.id
       INNER JOIN anime ON character_anime.anime_id = anime.id
       INNER JOIN author ON anime.author_id = author.id
WHERE (characters.`name` LIKE "%San%");
```

+ Filter animes by genre. For example, we want to list all the animes that is “Drama”:

```
SELECT title, episodes, series, premiered, start_on_air, finish_on_air,
ratings, anime.`status`, `source`, views, author.`name` as "Author name"
FROM Have
       INNER JOIN genre ON have.genre_id = genre.id
       INNER JOIN anime ON have.anime_id = anime.id
    INNER JOIN author ON anime.author_id = author.id
WHERE (genre.`name` LIKE "%Drama%");
```

+ Filter animes by studio. For example, we want to list all the animes that was made by “Ghibli:

```
SELECT title, episodes, series, premiered, start_on_air, finish_on_air,
ratings, anime.`status`, `source`, views, author.`name` as "Author name"
FROM Make
       INNER JOIN studio ON make.studio_id = studio.id
       INNER JOIN anime ON make.anime_id = anime.id
    INNER JOIN author ON anime.author_id = author.id
WHERE (studio.`name` LIKE "%Ghibli%");
```

+ Sort all animes by total views:

```
SELECT title, episodes, series, premiered, start_on_air, finish_on_air,
ratings, anime.`status`, `source`, views, author.`name` as "Author name"
FROM anime
       INNER JOIN author ON anime.author_id = author.id
ORDER BY views DESC;
```

+ Update the number of completed animes of watchers:

```
DROP TABLE IF EXISTS completed;
DROP PROCEDURE IF EXISTS insert_to_completed;
DROP PROCEDURE IF EXISTS update_completed;
DROP PROCEDURE IF EXISTS update_watcher;

DELIMITER $$
CREATE PROCEDURE insert_to_completed()
BEGIN
DECLARE no INT;
  SET no = 0;
  meow: LOOP
    SET no = no +1;
    INSERT INTO completed (watcher_id, total) VALUES (no, default);
    IF no =(SELECT MAX(watcher.id) FROM watcher) THEN
       LEAVE meow;
    END IF;
 END LOOP meow;
END $$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE update_completed()
BEGIN
DECLARE no INT;
  SET no = 0;
  meow: LOOP
    SET no = no +1;
    UPDATE completed
    SET total = IFNULL((SELECT COUNT(history.anime_id) FROM history
       INNER JOIN watcher ON history.watcher_id = watcher.id
       INNER JOIN episode_anime ON history.episode_id =
episode_anime.episode_id
       WHERE watcher_id = no
       HAVING MAX(history.episode_id) = MAX(episode_anime.episode_id)),0)
WHERE watcher_id = no;
    IF no = (SELECT MAX(watcher.id) FROM watcher) THEN
       LEAVE meow;
    END IF;
 END LOOP meow;
END $$
DELIMITER ;
```

```
DELIMITER $$
CREATE PROCEDURE update_watcher()
BEGIN
DECLARE no INT;
  SET no = 0;
  meow: LOOP
    SET no = no +1;
    UPDATE watcher
    SET completed = IFNULL((SELECT total FROM completed

        INNER JOIN watcher ON completed.watcher_id = watcher.id

        WHERE watcher_id = no),0)
                                                    WHERE id = no;
    IF no = (SELECT MAX(watcher.id) FROM watcher) THEN
        LEAVE meow;
    END IF;
 END LOOP meow;
END $$
DELIMITER ;

CREATE TABLE IF NOT EXISTS completed (
       watcher_id INT NOT NULL,
       total INT DEFAULT 0,
       FOREIGN KEY (watcher_id) REFERENCES watcher(id),
       PRIMARY KEY (watcher_id)
);
CALL insert_to_completed;
CALL update_completed;
CALL update_watcher;
```

+ View other watcher's account:

```
DELIMITER $$
CREATE PROCEDURE check_others_account(IN `name` VARCHAR(100))
BEGIN
       SELECT `name`, gender, dob, email, watching, plan_to_watch,
completed
    FROM watcher
       WHERE (watcher.`name` = `name`);
END $$
DELIMITER ;
```

+ View his/her own account:

```
DELIMITER $$
CREATE PROCEDURE check_my_account(IN `name` VARCHAR(100))
BEGIN
	SELECT `name`, gender, dob, email, watching, plan_to_watch,
completed
    FROM watcher
	WHERE (watcher.`name` = `name`);
END $$
DELIMITER ;
```

+ View his/her own watch history:

```
DELIMITER $$
CREATE PROCEDURE view_my_history(IN watcher_id VARCHAR(100))
BEGIN
	SELECT anime.title, episode.number_of_order as "episode",
`History`.`time` FROM `History`
	INNER JOIN watcher ON history.watcher_id = watcher.id
	INNER JOIN anime ON history.anime_id = anime.id
    INNER JOIN episode ON episode.id = history.episode_id
	WHERE `History`.watcher_id = watcher_id;
END $$
DELIMITER ;
```

+ View his/her own favorite animes:

```
DELIMITER $$
CREATE PROCEDURE view_my_favorite(IN watcher_id VARCHAR(100))
BEGIN
	SELECT anime.title FROM favorite
	INNER JOIN watcher ON favorite.watcher_id = watcher.id
	INNER JOIN anime ON favorite.anime_id = anime.id
	WHERE favorite.watcher_id = watcher_id
	GROUP BY anime.id;
END $$
DELIMITER ;
```

+ View his/her own “plan to watch”

```
DELIMITER $$
CREATE PROCEDURE view_my_plan(IN watcher_id VARCHAR(100))
BEGIN
	SELECT anime.title FROM plan_to_watch
	INNER JOIN watcher ON plan_to_watch.watcher_id = watcher.id
	INNER JOIN anime ON plan_to_watch.anime_id = anime.id
	WHERE plan_to_watch.watcher_id = watcher_id
	GROUP BY anime.id;
END $$
DELIMITER ;
```

These are all the functionalities provided to watchers.