Student: Joaquin Saldana
CS340 – Intro to Databases
Final Project – Game of Thrones Theme

Outline

Are you a fan of Game of Thrones? But have a hard time keeping track of all the characters? With a world, as large as that created by George R. R. Martin, it's can be difficult to keep track of all plot lines and characters involved.

This database intentions are to allow the user to enter a character and their relationships with houses, ancestral castles, titles they may hold, and their family relationships. We hope this allows the user to see the relationships amongst all the character's in the Game of Throne universe.

Database Outline

I have identified 6 entities that are essential to the database. Below is a brief description of the entities and their relationships and attributes with other entities as well any constraints to the relationships.

Characters

- Each character will have a unique primary key, an ID
- Each character will have a first name and last name, and the combination will be unique.
- A character may have been born in a castle
- A character may be affiliated with a house

Houses

- Each house will have a unique primary key, an ID
- Each house will have a unique house name
- A house may have a unique ancestral weapon
- A house may have an ancestral castle associated with it, this is many to one relationship. A house may only be associated with one ancestral castle, but an ancestral castle may be associated with more than one house (i.e. Kings Landing)

Ancestral Castles

- Each castle will have a unique primary key, an ID
- Each castle will have a unique castle name
- Each castle will have a location

Titles

- Each title will have a unique primary key, an ID
- Each title will have a unique title name

Character_Title

- A character may be associated to a title, and possibly more than one title
- A title may have been held by more than one character in the world of a song of ice and fire
- This table represents a many to many relationship

Family

- This is a recursive relationship reflected in a separate entity
- A character, if the information is available, will be associated to one father and one mother.
- A character may be a father or mother to more than one character
- This is a one to many relationship

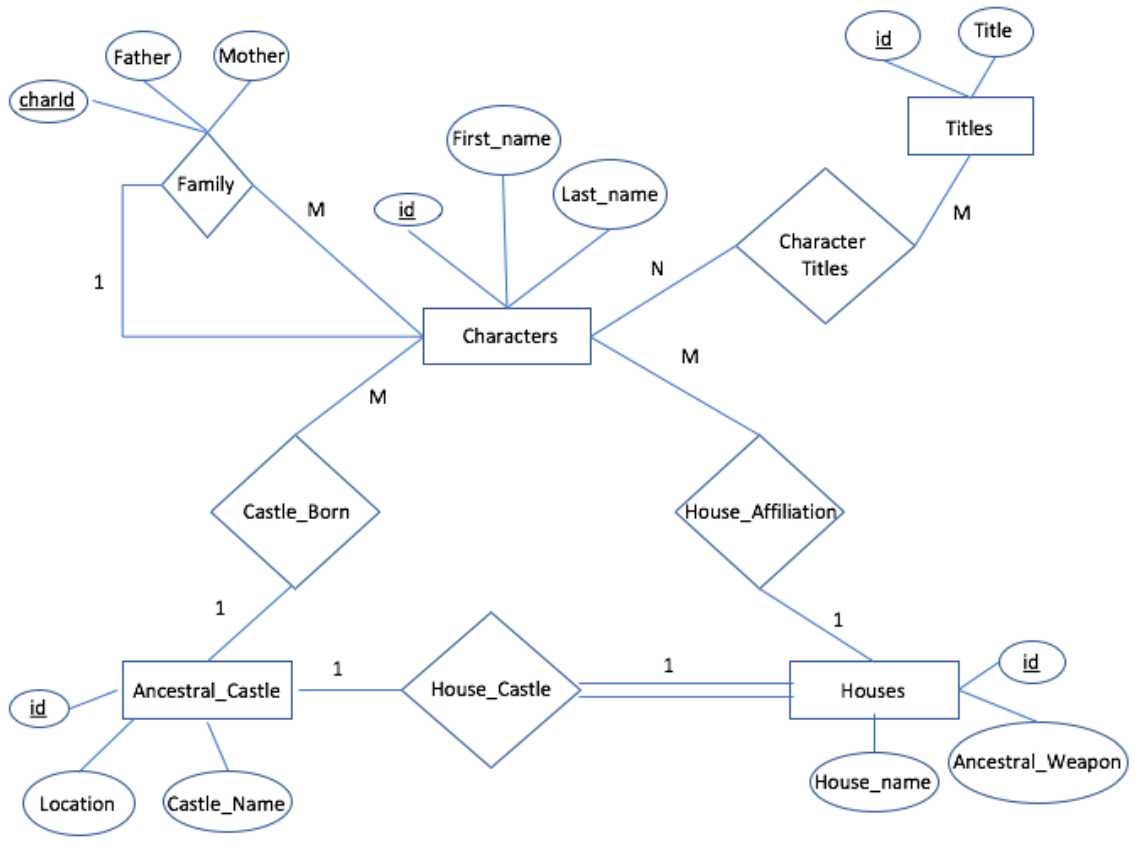
It should be noted that in some of the table modifications, it was necessary to for me ask the user for input in regards to updating and deleting. However, where possible, I allowed drop down select menus for the user to choose from data already entered.

In example, for the characters table, I felt it was easier for the user to enter the character's first name and last name they wish to modify or delete. I felt this was much more logically then providing the option of a drop down box with the characters name. However, in the same form I placed a drop down menu with the house affiliations already entered. If a house affiliation is not present, then the user should add it to the house table first THEN make the entry to the character's table.

****** IMPORTANT NOTE: modification or deletions that require user input via string, will require the string be an exact match to the data in the table for the action to be successful. ********

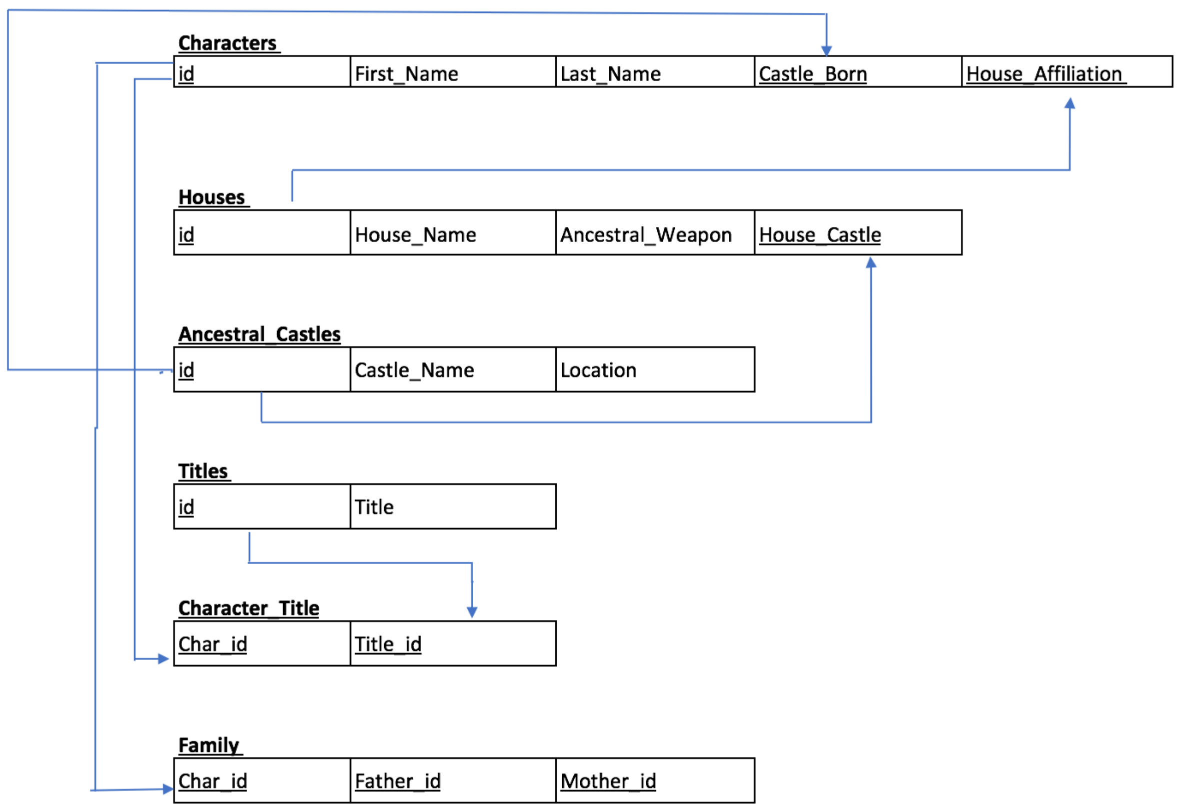
When a tuple is removed from a table, it is reflected as null in tables that reference the data.

ERM Diagram



Father
Mother
charId
Family
M
1
id
First_name
Last_name
id
Title
Titles
Characters
N
Character Titles
M
M
M
Castle_Born
House_Affiliation
1
1
Ancestral_Castle
1
House_Castle
1
Houses
id
id
Location
Castle_Name
House_name
Ancestral_Weapon

Schema



Data Definition Queries

-- Student: Joaquin Saldana (saldanaj@oregonstate.edu)

-- Drop the tables if they exists. The order in which the tables will be created
-- is in ascending order with the table with less foreign keys to the table
-- w/ the most foreign keys

```
DROP TABLE IF EXISTS `family`;
DROP TABLE IF EXISTS `character_title`;
DROP TABLE IF EXISTS `characters`;
DROP TABLE IF EXISTS `houses`;
DROP TABLE IF EXISTS `titles`;
DROP TABLE IF EXISTS `ancestral_castles`;
```

-- Create the ancestral_castles table that will contain the following
-- properties/attributes:

```
-- id: auto incrementing int which is the primary key
-- castle_name: varchar, max length 255, cannot be null
-- location: varchar, max length 255, cannot be null

CREATE TABLE ancestral_castles(
	id INT AUTO_INCREMENT NOT NULL,
	castle_name VARCHAR(255) NOT NULL,
	location VARCHAR(255) NOT NULL,
	PRIMARY KEY (id),
	UNIQUE (castle_name)
)ENGINE=InnoDB;


-- Create the houses table that will contain the following
-- properties/attributes:
-- id: auto incrementing int which is the primary key
-- house_name: varchar, max length 255, cannot be null
-- ancestral_weapon: varchar, max length 255
-- house_castle: a int which is a foreign key reference to the acentral_castle table

CREATE TABLE houses(
	id INT AUTO_INCREMENT NOT NULL,
	house_name VARCHAR(255) NOT NULL,
	ancestral_weapon VARCHAR(255),
	house_castle INT,
	UNIQUE(house_name),
	PRIMARY KEY (id),
	FOREIGN KEY (house_castle) REFERENCES ancestral_castles(id) ON UPDATE SET NULL
ON DELETE SET NULL
)ENGINE=InnoDB;

-- Create the titles table that will contain the following
-- properties/attributes:
-- id: auto incrementing int which is the primary key
-- title_name: varchar, max length 255, cannot be null

CREATE TABLE titles(
	id INT AUTO_INCREMENT NOT NULL,
	title_name VARCHAR(255) NOT NULL,
	PRIMARY KEY(id),
	UNIQUE(title_name)
)ENGINE=InnoDB;
```

```
-- Create the characters table that will contain the following
-- properties/attributes:
-- id: auto incrementing int which is the primary key
-- first_name: varchar, max length 255, cannot be null
-- last_name: varchar, max length 255, cannot be null
-- castle_born: int which is a foreign key reference to the ancestral_castles table
-- house_affiliation: int which is a foreign key reference to the houses table

CREATE TABLE characters(
	id INT AUTO_INCREMENT NOT NULL,
	first_name VARCHAR(255) NOT NULL,
	last_name VARCHAR(255) NOT NULL,
	castle_born INT,
	house_affiliation INT,
	PRIMARY KEY(id),
	FOREIGN KEY (castle_born) REFERENCES ancestral_castles(id) ON UPDATE SET NULL ON
DELETE SET NULL,
	FOREIGN KEY (house_affiliation) REFERENCES houses(id) ON UPDATE SET NULL ON
DELETE SET NULL,
	UNIQUE KEY (first_name, last_name)
)ENGINE=InnoDB;


-- Create the character_title table that will contain the following
-- properties/attributes:
-- char_id: int which is a foreign key reference to the characters id
-- title_id: int which is a foreign key reference to the title's id

CREATE TABLE character_title(
	char_id INT,
	title_id INT,
	FOREIGN KEY (char_id) REFERENCES characters(id) ON UPDATE SET NULL ON DELETE SET
NULL,
	FOREIGN KEY (title_id) REFERENCES titles(id) ON UPDATE SET NULL ON DELETE SET NULL
)ENGINE=InnoDB;

-- Create the family table that will contain the following
-- properties/attributes:
-- chard_id: int which is a foreign key reference to the characters id
-- father_id: int which is a foreign key reference to the characters id which is the father
-- mother_id: int which is a foreign key reference to the characters id which is the mother

CREATE TABLE family(
	char_id INT,
	father_id INT,
	mother_id INT,
	FOREIGN KEY (char_id) REFERENCES characters(id) ON UPDATE SET NULL ON DELETE SET
NULL,
	FOREIGN KEY (father_id) REFERENCES characters(id) ON UPDATE SET NULL ON DELETE
SET NULL,
	FOREIGN KEY (mother_id) REFERENCES characters(id) ON UPDATE SET NULL ON DELETE
SET NULL
)ENGINE=InnoDB;


-- INSERTION SECTION

-- this is the insertion stmt for the ancestral_castles table:

INSERT INTO ancestral_castles (castle_name, location) VALUES ("Casterly Rock",
"Westerlands"),
	("Winterfell", "North Westeros"),
	("Kings Landing", "Crownloands Westeros"),
	("Dragonstone", "Island of Dragonstone"),
	("Riverrun", "Riverlands Westeros"),
	("Castle Black", "The Wall"),
	("Pyke", "Iron Islands"),
	("Sunspear", "Dorne Westeros"),
	("Dreadfort", "North Westeros"),
	("Storms End", "Stormlands");


-- this is the insertion statement for the houses table:

INSERT INTO houses (house_name, ancestral_weapon, house_castle) VALUES ("House
Lannister", "Brightroar", (SELECT id FROM ancestral_castles WHERE castle_name =
"Casterly Rock")),
	("House Targaryen", "Dark Sister", (SELECT id FROM ancestral_castles WHERE
castle_name = "Kings Landing")),
	("House Stark", "Ice", (SELECT id FROM ancestral_castles WHERE castle_name =
"Winterfell")),
	("House Baratheon", NULL, (SELECT id FROM ancestral_castles WHERE castle_name =
"Dragonstone")),
	("Nights Watch", NULL, (SELECT id FROM ancestral_castles WHERE castle_name =
"Castle Black")),
	("House Greyjoy", NULL, (SELECT id FROM ancestral_castles WHERE castle_name =
"Pyke"));
```

```
-- this is the insertion statement for the titles table:

INSERT INTO titles (title_name) VALUES ("Lord of the Iron Islands"),
	("Lord of Winterfell"),
	("Prince of Winterfell"),
	("Protector of the Realm"),
	("Lady of Casterly Rock"),
	("Warden of the West"),
	("Warden of the North"),
	("Hand of the King"),
	("Warden of the South"),
	("Lord of Highgarden"),
	("Acting Hand of the King"),
	("Lord of Casterly Rock"),
	("King of Westeros"),
	("Lord of Dragonstone"),
	("The King of Dragonstone"),
	("Queen of the Seven Kingdoms"),
	("Khaleesi"),
	("Princess of Dragonstone"),
	("Lord of the Crossing"),
	("The Mad King");


-- this is the insertion statement for the characters table:

INSERT INTO characters(first_name, last_name, castle_born, house_affiliation) VALUES
	("Eddard", "Stark", (SELECT id FROM ancestral_castles WHERE castle_name =
"Winterfell"), (SELECT id FROM houses WHERE house_name = "House Stark")),
	("Cersei", "Lannister", (SELECT id FROM ancestral_castles WHERE castle_name =
"Casterly Rock"), (SELECT id FROM houses WHERE house_name = "House Lannister")),
	("Tywin", "Lannister", NULL, (SELECT id FROM houses WHERE house_name = "House
Lannister")),
	("Daenerys", "Targaryen", (SELECT id FROM ancestral_castles WHERE castle_name =
"Dragonstone"), (SELECT id FROM houses WHERE house_name = "House Targaryen")),
	("Joanna", "Lannister", (SELECT id FROM ancestral_castles WHERE castle_name =
"Casterly Rock"), (SELECT id FROM houses WHERE house_name = "House Targaryen")),
	("Catelyn", "Stark", (SELECT id FROM ancestral_castles WHERE castle_name =
"Riverrun"), (SELECT id FROM houses WHERE house_name = "House Stark")),
	("Arya", "Stark", (SELECT id FROM ancestral_castles WHERE castle_name = "Winterfell"),
(SELECT id FROM houses WHERE house_name = "House Stark")),
	("Stannis", "Baratheon", (SELECT id FROM ancestral_castles WHERE castle_name =
"Storms End"), (SELECT id FROM houses WHERE house_name = "House Baratheon")),
        ("Aerys II", "Targaryen", (SELECT id FROM ancestral_castles WHERE castle_name =
"Kings Landing"), (SELECT id FROM houses WHERE house_name = "House Targaryen")),
        ("Rhaella", "Targaryen", (SELECT id FROM ancestral_castles WHERE castle_name =
"Kings Landing"), (SELECT id FROM houses WHERE house_name = "House Targaryen")),
        ("Jon", "Snow", NULL, (SELECT id FROM houses WHERE house_name = "Nights Watch"));

-- this is the insertion statements for the character_title table:

INSERT INTO character_title(char_id, title_id) VALUES
        ((SELECT id FROM characters WHERE first_name = "Eddard" AND last_name = "Stark"),
(SELECT id FROM titles WHERE title_name = "Lord of Winterfell")),
        ((SELECT id FROM characters WHERE first_name = "Eddard" AND last_name = "Stark"),
(SELECT id FROM titles WHERE title_name = "Hand of the King")),
        ((SELECT id FROM characters WHERE first_name = "Eddard" AND last_name = "Stark"),
(SELECT id FROM titles WHERE title_name = "Warden of the North")),
        ((SELECT id FROM characters WHERE first_name = "Tywin" AND last_name =
"Lannister"), (SELECT id FROM titles WHERE title_name = "Hand of the King")),
        ((SELECT id FROM characters WHERE first_name = "Tywin" AND last_name =
"Lannister"), (SELECT id FROM titles WHERE title_name = "Lord of Casterly Rock")),
        ((SELECT id FROM characters WHERE first_name = "Daenerys" AND last_name =
"Targaryen"), (SELECT id FROM titles WHERE title_name = "Khaleesi")),
        ((SELECT id FROM characters WHERE first_name = "Daenerys" AND last_name =
"Targaryen"), (SELECT id FROM titles WHERE title_name = "Queen of the Seven Kingdoms")),
        ((SELECT id FROM characters WHERE first_name = "Aerys II" AND last_name =
"Targaryen"), (SELECT id FROM titles WHERE title_name = "The Mad King")),
        ((SELECT id FROM characters WHERE first_name = "Aerys II" AND last_name =
"Targaryen"), (SELECT id FROM titles WHERE title_name = "King of Westeros")),
        ((SELECT id FROM characters WHERE first_name = "Aerys II" AND last_name =
"Targaryen"), (SELECT id FROM titles WHERE title_name = "Protector of the Realm")),
        ((SELECT id FROM characters WHERE first_name = "Eddard" AND last_name = "Stark"),
(SELECT id FROM titles WHERE title_name = "Protector of the Realm"));


-- this is the insertion statements for the family table:
INSERT INTO family (char_id, father_id, mother_id) VALUES
        ((SELECT id FROM characters WHERE first_name = "Arya" AND last_name = "Stark"),
(SELECT id FROM characters WHERE first_name = "Eddard" AND last_name = "Stark"), (SELECT
id FROM characters WHERE first_name = "Catelyn" AND last_name = "Stark")),
        ((SELECT id FROM characters WHERE first_name = "Daenerys" AND last_name =
"Targaryen"), (SELECT id FROM characters WHERE first_name = "Aerys II" AND last_name =
"Targaryen"), (SELECT id FROM characters WHERE first_name = "Rhaella" AND last_name =
"Targaryen")),
        ((SELECT id FROM characters WHERE first_name = "Cersei" AND last_name =
"Lannister"), (SELECT id FROM characters WHERE first_name = "Tywin" AND last_name =
"Lannister"), (SELECT id FROM characters WHERE first_name = "Joanna" AND last_name =
"Lannister"));
```

Data Manipulation Queries

/* Queries to retrieve the character’s table foreign key data */

```
// this query performs two left joins so we can pull what the foreign keys in the
// character’s table reference which are ancestral castles table and the houses
// table

SELECT characters.first_name, characters.last_name, ancestral_castles.castle_name,
houses.house_name FROM characters LEFT JOIN ancestral_castles ON characters.castle_born =
ancestral_castles.id LEFT JOIN houses ON characters.house_affiliation = houses.id

SELECT id, castle_name FROM ancestral_castles

SELECT id, house_name FROM houses
```

/* Queries to modify the character tables information*/

```
INSERT INTO characters(first_name, last_name, house_affiliation) VALUES [firstNameInput],
[lastNameInput], [houseAffiliatedInput]

INSERT INTO characters(first_name, last_name, castle_born) VALUES [firstNameInput],
[lastNameInput], [castleBornInput]

INSERT INTO characters(first_name, last_name) VALUES [firstNameInput], [lastNameInput],

INSERT INTO characters(first_name, last_name, castle_born, house_affiliation) VALUES
[firstNameInput], [lastNameInput], [castleBornInput] , [houseAffiliatedInput]

UPDATE characters SET castle_born = NULL, house_affiliation = [houseAffiliatedInput] WHERE
first_name = [firstNameInput]AND last_name = [lastNameInput]

UPDATE characters SET castle_born = [castleBornInput], house_affiliation = NULL WHERE
first_name = [firstNameInput] AND last_name = [lastNameInput]

UPDATE characters SET castle_born = NULL, house_affiliation = NULL WHERE first_name =
[firstNameInput] AND last_name = [lastNameInput]

UPDATE characters SET castle_born =[castleBornInput] , house_affiliation =
[houseAffiliationINput] WHERE first_name = [firstNameInput] AND last_name =
[lastNameInput]
```

DELETE FROM characters WHERE first_name = [firstNameInput] AND last_name = [lastNameInput]

/* Queries to retrieve the houses table foreign key data */

SELECT houses.house_name, houses.ancestral_weapon, ancestral_castles.castle_name FROM houses LEFT JOIN ancestral_castles ON houses.house_castle = ancestral_castles.id

SELECT id, castle_name FROM ancestral_castles

/* Queries to modify the houses tables information via forms*/

INSERT INTO houses(house_name, ancestral_weapon) VALUES [houseNameInput], [weaponInput]

INSERT INTO houses(house_name, ancestral_weapon, house_castle) VALUES [houseNameINput], [weaponInput], [houseCastleInput]

UPDATE houses SET ancestral_weapon = [weaponInput], house_castle = NULL WHERE house_name = [houseNameInput]

UPDATE houses SET ancestral_weapon = [weaponInput], house_castle = [houseCastleInput] WHERE house_name = [houseNameinput]

DELETE FROM houses WHERE house_name = [houseNameInput]

/* Queries to retrieve the ancestral castle table data*/

SELECT castle_name, location FROM ancestral_castles

/* Queries to modify the ancestral castles tables information via forms*/

INSERT INTO ancestral_castles(castle_name, location) VALUES [castleinput], [locationInput]

UPDATE ancestral_castles SET location = [locationInput] WHERE castle_name = [castleNameInput]

DELETE FROM ancestral_castles WHERE castle_name = [castleNameInput]

/* Queries to retrieve the data in the titles table*/

SELECT title_name FROM titles

/* Queries to modify the titles tables information via forms*/

INSERT INTO titles(title_name) VALUES [titleNameInput]

DELETE FROM titles WHERE title_name = [titleNameInput]

/* Queries to retrieve the data in the character titles table*/

SELECT characters.first_name, characters.last_name, titles.title_name FROM character_title LEFT JOIN characters ON character_title.char_id = characters.id LEFT JOIN titles ON titles.id = character_title.title_id

SELECT id, title_name FROM titles

/* Queries to modify the character titles tables information via forms*/

INSERT INTO character_title(char_id, title_id) VALUES ((SELECT id FROM characters WHERE first_name = [firstNameInput] AND last_name = [lastNameInput]) ,[titleIDInput])

DELETE FROM character_title WHERE char_id = (SELECT id FROM characters WHERE first_name = [firstNameInput] AND last_name = [lastNameInput]) AND title_id = [titleIDInput]

/* Queries to retrieve the data in the family table*/

// this is a query where we have to left join the character's table
// three times so to get the childs first/last name, the fathers first/last name
// the mothers first/last name

SELECT child.first_name, child.last_name, father.first_name, father.last_name, mother.first_name, mother.last_name FROM family LEFT JOIN characters AS child ON family.char_id = child.id LEFT JOIN characters AS father ON family.father_id = father.id LEFT JOIN characters AS mother ON family.mother_id = mother.id

SELECT id, first_name, last_name FROM characters

/* Queries to modify the family tables information via forms*/

INSERT INTO family(char_id, father_id, mother_id) VALUES [childIDInput], [fatherIDInput], [motherIDInput]

DELETE FROM family WHERE char_id = [childIDInput]

/* Queries to retrieve the data for the filter of the character titles */

```
// this query is used to populate the dynamic filter selection to filter by the characters
// name, to avoid the same name appearing more than once, we group by the character id
// this will only return the character whom have a title associated with themselves

SELECT characters.id, characters.first_name, characters.last_name FROM character_title
LEFT JOIN characters ON characters.id = character_title.char_id
GROUP BY characters.id


// this query is to dynamically populate the selection list for the titles to filter by
// this will only return the name of titles that have been associated to a character

SELECT titles.id, titles.title_name FROM character_title LEFT JOIN titles ON titles.id =
character_title.title_id GROUP BY titles.id
```

/* Queries to populate the table returned by the filter*/

```
// this query returns the table when the user filters by the title

SELECT queryOne.first_name, queryOne.last_name, queryOne.title_name FROM (SELECT
character_title.char_id, character_title.title_id, characters.first_name, characters.last_name,
titles.title_name FROM character_title
LEFT JOIN characters ON characters.id = character_title.char_id
LEFT JOIN titles  ON titles.id = character_title.title_id) AS queryOne
WHERE queryOne.title_id = [titleNumberSelectedInFilter]

// this query returns the table when the user filters by the character

SELECT queryOne.first_name, queryOne.last_name, queryOne.title_name FROM (SELECT
character_title.char_id, character_title.title_id, characters.first_name, characters.last_name,
titles.title_name FROM character_title
LEFT JOIN characters ON characters.id = character_title.char_id
LEFT JOIN titles  ON titles.id = character_title.title_id) AS queryOne
WHERE queryOne.char_id = [characterIDSelectedInFilter]
```