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

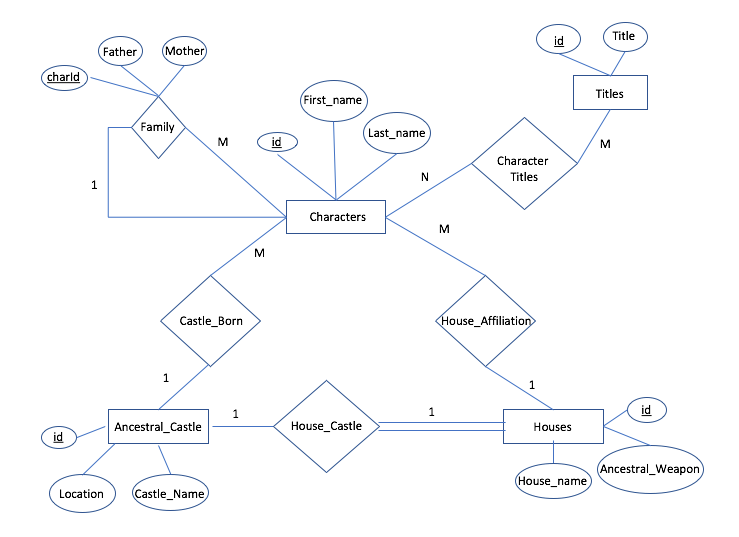
It should be noted that in some of the table modifications, it was necessary to for me ask the user for input in regards to updating and deleting. However, where possible, I allowed drop down select menus for the user to choose from data already entered.

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.

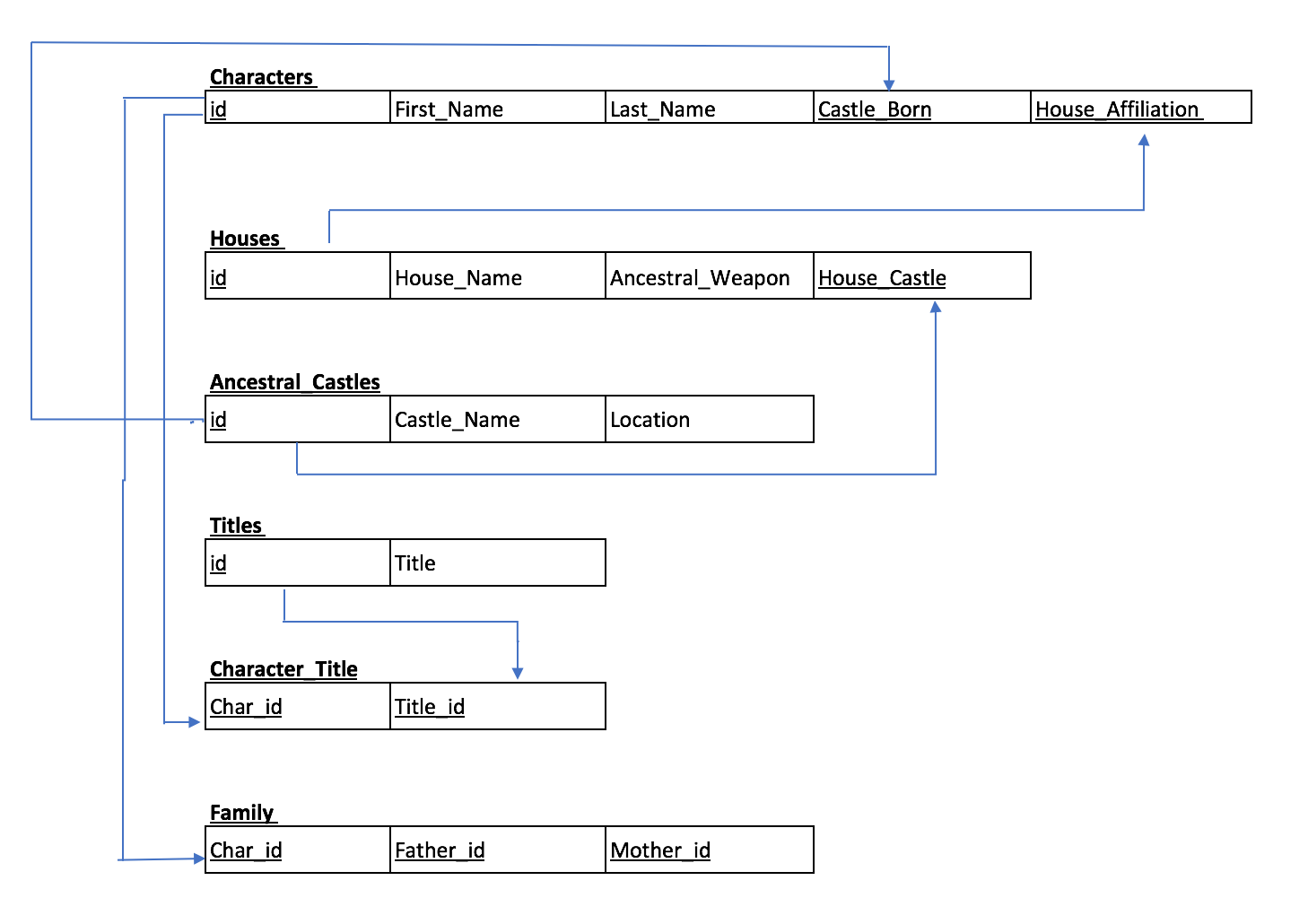
\*\*\*\*\*\* IMPORTANT NOTE: modification or deletions that require user input via string, will require the string be an exact match to the data in the table for the action to be successful. \*\*\*\*\*\*\*\*

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

**ERM Diagram**



**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]