Bonnie McNeil and Jordan Alhadeff

CS340 Final Project

**Topic Overview**

We have created a database-backed website modeling the Mortal Kombat universe. This is from a video game series featuring combatants (“Kharacters”) defending their Realm against invaders from other Realms. As video gaming has advanced, so has storytelling in the medium and now much more is known about the Kharacters than ever before. The universe, while Kharacter centric, has expanded in recent years to include more information about each Kharacter including a Kharacter’s origin Realm, Faction affiliation, and Species.

**Database Outline**

**Entities**

Kharacter

Attributes: ID (Primary Key), First Name, Last Name, Gender (Male or Female), Species (Foreign Key), Realm (Foreign Key), Faction (Foreign Key), Living (Yes or No), and Alignment (Good, Bad, or Neutral).

Relationships:

In: Some Kharacters are In Factions, referenced by a foreign key to the Faction.ID. Factions can contain many Kharacters (or none)

Are: All Kharacters Are a Species, referenced by a foreign key to Species.ID. Species can have many Kharacters (or none)

From: All Kharacters are From a Realm, referenced by a foreign key to Realm.ID. Realms may contain many Kharacters (or none)

Faction

Attributes: ID (Primary Key), Name (Unique), Active (Yes or No), Leader (Foreign Key)

Relationships:

Leader: Most factions have a leader, which is a Kharacter referenced by a foreign key to Kharacter.ID.  A Faction can only have one leader (Kharacter) and a leader can only lead one faction. One to many or one to one relationship?

Species

Attributes: ID (Primary Key), Name (Unique), Realm of origin (Foreign Key)

Relationships:

Origin: All Species have a Realm of Origin, referenced by a foreign key to Realm.ID. Realms can have many species that originate from them

Realm

Attributes: ID (Primary Key), Name (Unique), Native Species (Foreign Key), Conquered (Yes or No)

Relationships:

Native Species:  A realm can have one to many species originate from it.  Native    Species is a foreign key reference to Species Id.

Faction\_Species table: Many-to-many relationship Have

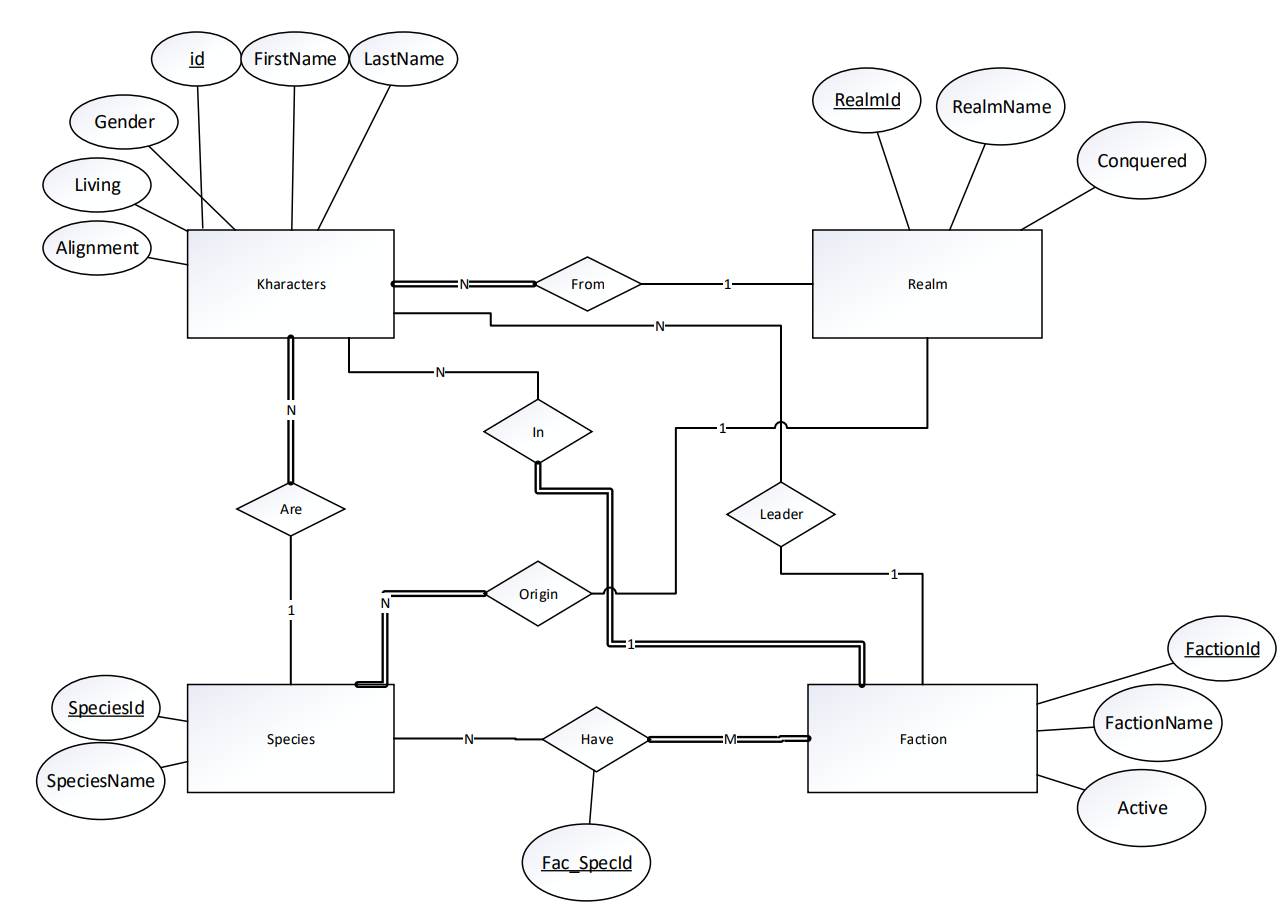
All Factions Have founding Species. Factions are founded by one or more Species. Species can found many Factions or none. This is a many-to-many relationship, referenced in the table Faction\_Species with foreign keys to Faction.ID and Species.ID

Faction\_Species

Attributes: Faction\_Species Id, FId (Foreign Key), SId (Foreign Key) (FId and SId must be Unique)

Relationships: This many to many table captures that a Faction can have many species in it and a species can be affiliated with many factions.  FId is a foreign key reference to to Faction Id while SId is a foreign key reference to Species Id.

**ER Diagram**



**Schema**

**mk\_Kharacters**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **FirstName** | **LastName** | **Gender** | **Species** | **Realm** | **Faction** | **Living** | **Alignment** |

**mk\_Species**

|  |  |  |
| --- | --- | --- |
| **SpeciesId** | **SpeciesName** | **RealmOrigin** |

**mk\_Realm**

|  |  |  |  |
| --- | --- | --- | --- |
| **RealmId** | **RealmName** | **NativeSpecies** | **Conquered** |

**mk\_Faction**

|  |  |  |  |
| --- | --- | --- | --- |
| **FactionId** | **FactionName** | **Active** | **Leader** |

**mk\_Faction\_Species**

|  |  |  |
| --- | --- | --- |
| **Fac\_SpecId** | **FId** | **SId** |

**Data Definition Queries**

CREATE TABLE `mk\_Kharacters`(

`id` Int(11) AUTO\_INCREMENT NOT NULL,

`FirstName` VARCHAR(255) NOT NULL,

`LastName` VARCHAR(255),

`Gender` VARCHAR(255),

`Species` INT(11),

`Realm` INT(11),

`Faction` INT(11),

`Living` VARCHAR(255),

`Alignment` VARCHAR(255),

PRIMARY KEY (`id`),

FOREIGN KEY (`Species`) REFERENCES `mk\_Species` (`SpeciesId`) ON UPDATE SET NULL ON DELETE SET NULL,

FOREIGN KEY (`Realm`) REFERENCES `mk\_Realm` (`RealmId`),

FOREIGN KEY (`Faction`) REFERENCES `mk\_Faction` (`FactionId`)

) ENGINE=InnoDB;

CREATE TABLE `mk\_Species`(

`SpeciesId` INT(11) AUTO\_INCREMENT NOT NULL,

`SpeciesName` VARCHAR(255) NOT NULL,

`RealmOrigin` INT(11),

PRIMARY KEY (`SpeciesId`),

UNIQUE (`SpeciesName`),

FOREIGN KEY (`RealmOrigin`) REFERENCES `mk\_Realm` (`RealmId`)

) Engine=INNODB;

CREATE TABLE `mk\_Realm`(

`RealmId` INT(11) AUTO\_INCREMENT NOT NULL,

`RealmName` VARCHAR(255) NOT NULL,

`NativeSpecies` INT(11),

`Conquered` VARCHAR(255),

PRIMARY KEY (`RealmId`),

UNIQUE (`RealmName`),

FOREIGN KEY (`NativeSpecies`) REFERENCES `mk\_Species` (`SpeciesId`) ON UPDATE SET NULL ON DELETE SET NULL

)Engine=INNODB;

CREATE TABLE `mk\_Faction`(

`FactionId` INT(11) AUTO\_INCREMENT NOT NULL,

`FactionName` VARCHAR(255) NOT NULL,

`Active` VARCHAR(255),

`Leader` INT(11),

PRIMARY KEY (`FactionId`),

UNIQUE (`FactionName`),

FOREIGN KEY (`Leader`) REFERENCES `mk\_Kharacters` (`id`) ON UPDATE SET NULL ON DELETE SET NULL

)ENGINE=INNODB;

CREATE TABLE `mk\_Faction\_Species`(

`Fac\_SpecId` INT(11) AUTO\_INCREMENT NOT NULL,

`FId` INT(11) NOT NULL,

`SId` INT(11) NOT NULL,

PRIMARY KEY (`Fac\_SpecId`),

UNIQUE `fac\_spec` (`FId`,`SId`),

FOREIGN KEY (`FId`) REFERENCES `mk\_Faction` (`FactionId`) ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY (`SId`) REFERENCES `mk\_Species` (`SpeciesId`) ON UPDATE CASCADE ON DELETE CASCADE

)ENGINE=INNODB;

**Data Manipulation Queries**

Show all Kharacters:

SELECT mk\_Kharacters.id, mk\_Kharacters.FirstName, mk\_Kharacters.LastName, mk\_Kharacters.Gender, mk\_Species.SpeciesName AS Species, mk\_Realm.RealmName AS Realm, mk\_Faction.FactionName AS Faction, mk\_Kharacters.Living, mk\_Kharacters.Alignment FROM mk\_Kharacters LEFT JOIN mk\_Species ON mk\_Kharacters.Species=mk\_Species.SpeciesId LEFT JOIN mk\_Realm ON mk\_Kharacters.Realm=mk\_Realm.RealmId LEFT JOIN mk\_Faction ON mk\_Kharacters.Faction=mk\_Faction.FactionId

Show Kharacters by Alignment:

SELECT mk\_Kharacters.FirstName, mk\_Kharacters.LastName, mk\_Kharacters.Gender, mk\_Species.SpeciesName AS Species, mk\_Realm.RealmName AS Realm, mk\_Faction.FactionName AS Faction, mk\_Kharacters.Living, mk\_Kharacters.Alignment FROM mk\_Kharacters LEFT JOIN mk\_Species ON mk\_Kharacters.Species=mk\_Species.SpeciesId LEFT JOIN mk\_Realm ON mk\_Kharacters.Realm=mk\_Realm.RealmId LEFT JOIN mk\_Faction ON mk\_Kharacters.Faction=mk\_Faction.FactionId WHERE Alignment=[alignmentinput]

Show Kharacters by Faction:

SELECT mk\_Kharacters.FirstName, mk\_Kharacters.LastName, mk\_Kharacters.Gender, mk\_Species.SpeciesName AS Species, mk\_Realm.RealmName AS Realm, mk\_Faction.FactionName AS Faction, mk\_Kharacters.Living, mk\_Kharacters.Alignment FROM mk\_Kharacters LEFT JOIN mk\_Species ON mk\_Kharacters.Species=mk\_Species.SpeciesId LEFT JOIN mk\_Realm ON mk\_Kharacters.Realm=mk\_Realm.RealmId LEFT JOIN mk\_Faction ON mk\_Kharacters.Faction=mk\_Faction.FactionId WHERE mk\_Faction.FactionId=[factioninput]

Show Kharacters by Realm Conquered Status:

SELECT mk\_Kharacters.FirstName, mk\_Kharacters.LastName, mk\_Kharacters.Gender, mk\_Species.SpeciesName AS Species, mk\_Realm.RealmName AS Realm, mk\_Faction.FactionName AS Faction, mk\_Kharacters.Living, mk\_Kharacters.Alignment FROM mk\_Kharacters LEFT JOIN mk\_Species ON mk\_Kharacters.Species=mk\_Species.SpeciesId LEFT JOIN mk\_Realm ON mk\_Kharacters.Realm=mk\_Realm.RealmId LEFT JOIN mk\_Faction ON mk\_Kharacters.Faction=mk\_Faction.FactionId WHERE mk\_Realm.Conquered=[conqueredinput]

Select All Species:

SELECT mk\_Species.SpeciesId, mk\_Species.SpeciesName, mk\_Realm.RealmName FROM mk\_Species LEFT JOIN mk\_Realm ON mk\_Species.RealmOrigin=mk\_Realm.RealmId;

Select All Realms:

SELECT mk\_Realm.RealmId, mk\_Realm.RealmName, mk\_Species.SpeciesName, mk\_Realm.Conquered FROM mk\_Realm LEFT JOIN mk\_Species ON mk\_Realm.NativeSpecies=mk\_Species.SpeciesId;

Select All Factions:

SELECT mk\_Faction.FactionId, mk\_Faction.FactionName, mk\_Faction.Active, CONCAT(mk\_Kharacters.FirstName, " ", mk\_Kharacters.LastName) AS LeaderName FROM mk\_Faction LEFT JOIN mk\_Kharacters ON mk\_Faction.Leader=mk\_Kharacters.id;

Select Faction & Species Relationship:

SELECT mk\_Faction\_Species.Fac\_SpecId, mk\_Faction.FactionName AS RelFactionName, mk\_Species.SpeciesName AS RelSpeciesName FROM mk\_Faction\_Species LEFT JOIN mk\_Faction ON mk\_Faction.FactionId=mk\_Faction\_Species.FId LEFT JOIN mk\_Species ON mk\_Species.SpeciesId=mk\_Faction\_Species.SId;

Get Species by Faction:

SELECT mk\_Faction\_Species.Fac\_SpecId, mk\_Faction.FactionName AS RelFactionName, mk\_Species.SpeciesName AS RelSpeciesName FROM mk\_Faction\_Species LEFT JOIN mk\_Faction ON mk\_Faction.FactionId=mk\_Faction\_Species.FId LEFT JOIN mk\_Species ON mk\_Species.SpeciesId=mk\_Faction\_Species.SId WHERE mk\_Faction.FactionId=[factioninput]

Get Species:

SELECT SpeciesId, SpeciesName FROM mk\_Species;

Get Realms:

SELECT RealmId, RealmName FROM mk\_Realm;

Get Faction:

SELECT FactionId, FactionName FROM mk\_Faction;

Get Alignment:

SELECT DISTINCT Alignment FROM mk\_Kharacters;

Get Leader:

SELECT id, CONCAT (FirstName, " ", LastName) AS Leader FROM mk\_Kharacters WHERE mk\_Kharacters.id NOT IN (SELECT mk\_Faction.Leader FROM mk\_Faction WHERE mk\_Faction.Leader IS NOT NULL);

Faction Search:

SELECT mk\_Faction.FactionName, mk\_Faction.Active, CONCAT(mk\_Kharacters.FirstName, " ", mk\_Kharacters.LastName) AS LeaderName FROM mk\_Faction LEFT JOIN mk\_Kharacters ON mk\_Kharacters.id=mk\_Faction.Leader WHERE mk\_Faction.FactionName LIKE [factioninput]

Insert Kharacter:

INSERT INTO mk\_Kharacters SET [firstnameinput], [lastnameinput], [genderinput], [speciesinput], [realminput], [factioninput], [livinginput], [alignmentinput]

Insert Species:

INSERT INTO mk\_Species SET [speciesnameinput], [realmorigininput]

Insert Realm:

INSERT INTO mk\_Realm SET [realmnameinput], [nativespeciesinput], [conquered]

Insert Faction:

INSERT INTO mk\_Faction SET [factionnameinput], [activeinput], [leaderinput]

Insert Species Faction Relationship:

INSERT INTO mk\_Faction\_Species SET [factionname(id)input], [speciesname(id)input]

Delete Kharacter:

DELETE FROM mk\_Kharacters WHERE id=[character(id)input]

Delete Faction Species Relationship:

DELETE FROM mk\_Faction\_Species WHERE Fac\_SpecId=[faction/species(id)input]

Delete Species:

DELETE FROM mk\_Species WHERE SpeciesId=[species(id)input]

Edit Kharacter:

SELECT mk\_Kharacters.id, mk\_Kharacters.FirstName, mk\_Kharacters.LastName, mk\_Kharacters.Gender, mk\_Species.SpeciesName AS Species, mk\_Species.SpeciesId, mk\_Realm.RealmName AS Realm, mk\_Realm.RealmId, mk\_Faction.FactionName AS Faction, mk\_Faction.FactionId, mk\_Kharacters.Living, mk\_Kharacters.Alignment FROM mk\_Kharacters LEFT JOIN mk\_Species ON mk\_Kharacters.Species=mk\_Species.SpeciesId LEFT JOIN mk\_Realm ON mk\_Kharacters.Realm=mk\_Realm.RealmId LEFT JOIN mk\_Faction ON mk\_Kharacters.Faction=mk\_Faction.FactionId WHERE id=[kharacter(id)input]

Edit Species:

SELECT mk\_Species.SpeciesId, mk\_Species.SpeciesName, mk\_Species.RealmOrigin, mk\_Realm.RealmName FROM mk\_Species LEFT JOIN mk\_Realm ON mk\_Species.RealmOrigin=mk\_Realm.RealmId WHERE SpeciesId=[species(id)input]

Edit Realm:

SELECT mk\_Realm.RealmId, mk\_Realm.RealmName, mk\_Realm.NativeSpecies, mk\_Species.SpeciesName, mk\_Realm.Conquered FROM mk\_Realm LEFT JOIN mk\_Species ON mk\_Realm.NativeSpecies=mk\_Species.SpeciesId WHERE RealmId=[realm(id)input]

Edit Faction:

SELECT mk\_Faction.FactionId, mk\_Faction.FactionName, mk\_Faction.Active, mk\_Faction.Leader, CONCAT(mk\_Kharacters.FirstName, " ", mk\_Kharacters.LastName) AS LeaderName FROM mk\_Faction LEFT JOIN mk\_Kharacters ON mk\_Faction.Leader = mk\_Kharacters.id WHERE FactionId=[faction(id)input]

Update Kharacter:

UPDATE mk\_Kharacters SET FirstName=?, LastName=?, Gender=?, Species=?, Realm=?, Faction=?, Living=?, Alignment=? WHERE id=?[firstnameinput], [lastnameinput], [genderinput], [speciesinput], [realminput], [factioninput], [livinginput], [alignmentinput], [id(user selected kharacter to edit)input]

Update Species:

UPDATE mk\_Species SET SpeciesName=?, RealmOrigin=? WHERE SpeciesId=? [speciesnameinput], [realmorigininput], [species(user selected species to edit)input]

Update Realm:

UPDATE mk\_Realm SET RealmName=?, NativeSpecies=?, Conquered=? WHERE RealmId=? [realmnameinput], [nativespeciesinput], [conquered], [realm(user selected realm to edit)input]

Update Faction:

UPDATE mk\_Faction SET FactionName=?, Active=?, Leader=? WHERE FactionId=? [factionnameinput], [activeinput], [leaderinput], [faction(user selected faction to edit)input]

**Website Functionality**

The website allows the user to view all attributes for each entity table and relationship table. Users can insert data into each entity table. Users can also add to the many to many relationship by selecting a faction and adding a species relationship. Users can also remove a many to many relationship by selecting the Faction they want to remove a relationship then removing each Species relationship individually. Each entity table can be edited allowing the user to set any relationship to null, even if it does not make sense in the Mortal Kombat universe. Attributes requiring a user to create a relationship are built dynamically with the foreign key value not being exposed to the user. Special consideration was made to the Leader drop down so a leader to another faction does not show in that drop down of Kharacters. Does not make sense to have one Kharacter lead more than one Faction The user can delete from the Species and Kharacter entity tables. If a Kharacter is deleted, the corresponding Leader is set to null in the Faction table. If a Species is deleted, the corresponding Native Species from the Realm table and Species attribute in Kharacter table are set to null. In the Faction\_Species table, a removed species deletes the corresponding row. Species, Realm, and Faction names are all unique so the user cannot enter a duplicate. Furthermore the user cannot enter a duplicate Faction\_Species relationship. The user can filter Kharacters by Alignment, filter characters by affiliated Faction, filter Kharacters by whether their Realm was conquered, and search for a Faction.