Travis Robinson
Chelsea Colvin
CS 340
Spring 2016
Project URL: http://web.engr.oregonstate.edu/~robitrav/cs340/project.php

Fandoms Database

Outline

A Fandom is a term used to describe either a community of fans or the object of the fans' collective focus. For films, television shows, books, and games, these Fandoms can encompass fictional universes described in said mediums. Our Database is about various fictional universes as described by different Fandoms. With our Database, we will be tracking the Story of each Fandom, and the Genre that the Story is in. Within each Story, there will also be Characters which we will also be tracking. Characters come in many different types, which are described by the Characters Archetype(s) and their Origin Story.

This would be an interesting Database because it allows us to compare the characters and genres contained within each of these different Fandoms. These comparisons will help reveal patterns such as does a particular Archetype show up in a particular Genre more often, or does a Story have a lot of one type of Origin Story.

Database Outline in Words

Each Fandom Universe has a Story. Each Story must have an ID and Title. Each Story must have only one Genre.

Genre is the type of Story that a Fandom can be, such as Science Fiction or Fantasy. Each Story must have one Genre, but each Genre can be possessed by multiple stories. Each Genre has an ID and a Type.

CharacterArchetype is the type of person that a Character (contained in CharacterData) can be, such as Hero or Female. Each CharacterArchetype has an ID and a Type, and can be possessed by multiple characters.

CharacterData is the data of a Character residing in the Fandom Universe, as described in the associated Story. Each CharacterData must have an ID, First Name, and Origin Story. Characters can also have a Last Name. All characters have at least one Archetype, but may have more than one Archetype (such as Token Female and Naive Hero). Each Character is in only one Story, though each Story can have multiple Characters.

OriginStory is how a person came to be a Character in the Story. Each OriginStory has an ID and a Type. Each Origin Story can be held by any number of Characters (Characters being contained in CharacterData).

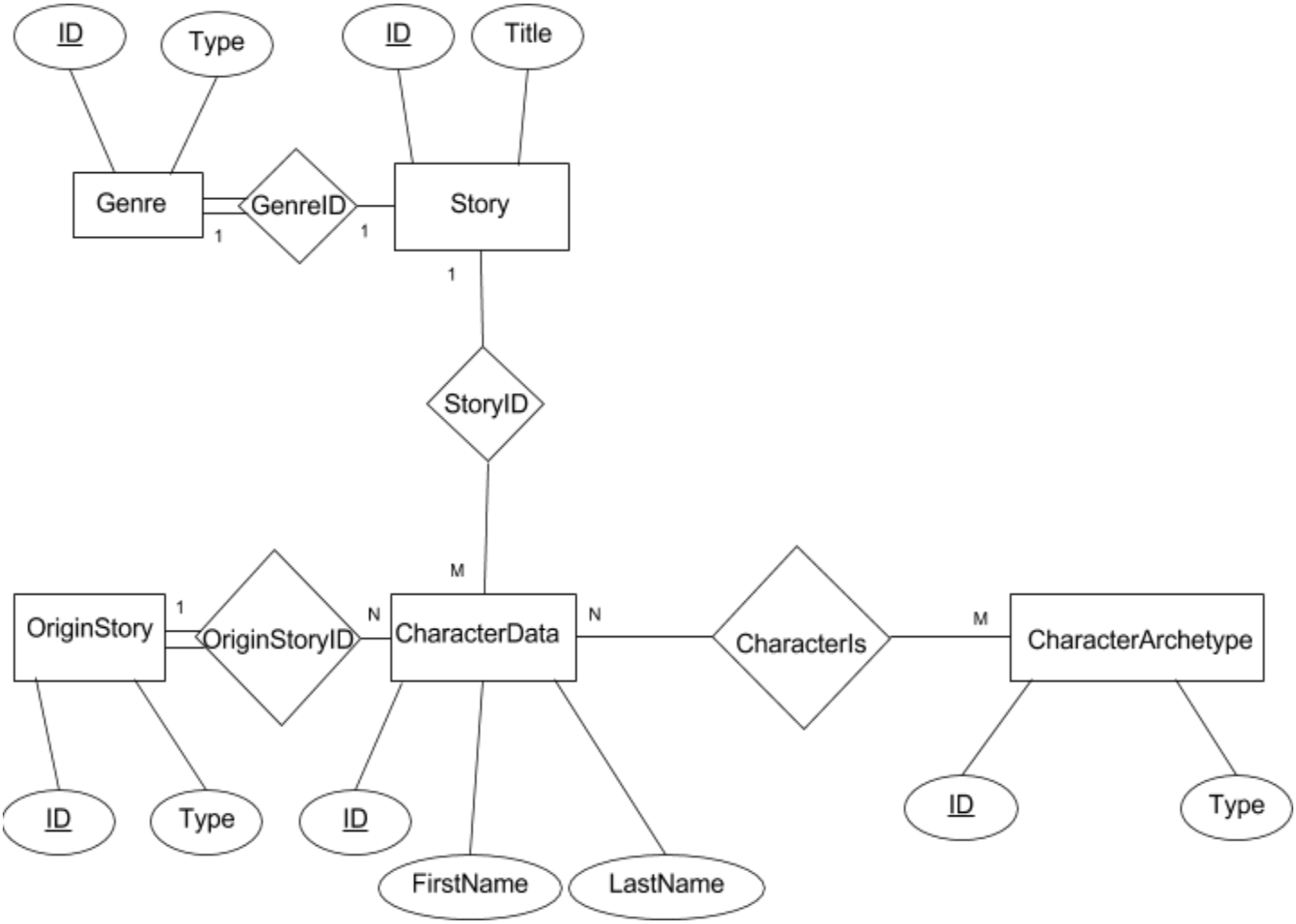
The entities of our database:

- Story - stories are fictional universes wherein the characters reside (such as Firefly or the Hunger Games)
- Character - characters have a name and archetype (i.e. Luke is a naive hero)
- Character Archetype-what type of character they are (i.e. token female like Leia)
- Genre - The genre that the story is (i.e. sci-fi, fantasy)
- Origin Story - The path the character took to becoming a hero (or villain), such as tragic beginning or a mentor.

The relationships between these entities:

- Characters can be multiple archetypes, and each archetype can be held by multiple characters-a many-to-many relationship
- Characters all have an origin story, but each character can only have one, while each origin story can be held by multiple characters-making a many to one relationship
- Each story has one genre, for a one-to-one relationship
- Each character is only in one story, while each story can hold multiple characters, for a one-to-many relationship

ER Diagram of Database



Database Schema

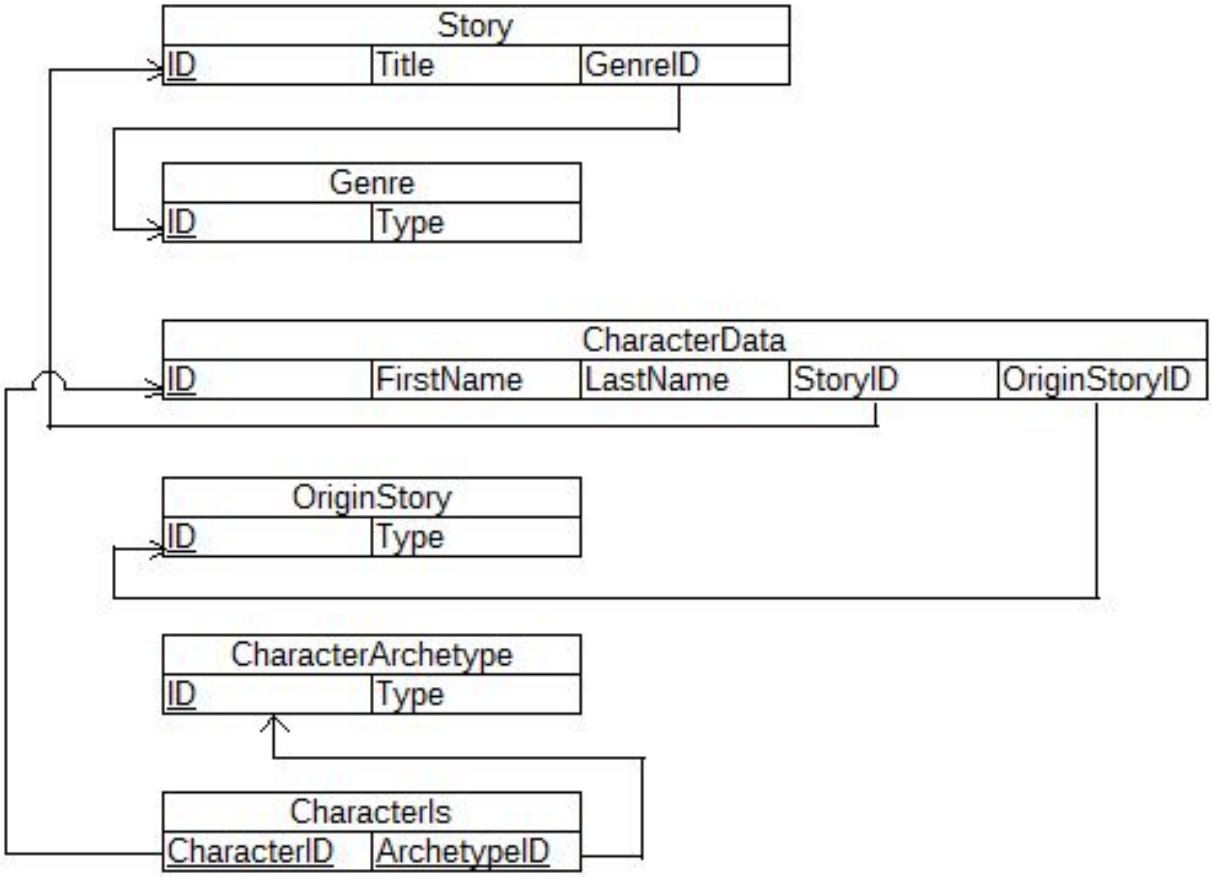


Table Creation Queries

I want to see the queries your ran to build your tables. These should not be in any of the website code because you should not be dynamically building or deleting tables.

```
DROP TABLE `CharacterIs`;
DROP TABLE `CharacterArchetype`;
DROP TABLE `CharacterData`;
DROP TABLE `Story`;
DROP TABLE `Genre`;
DROP TABLE `OriginStory`;
DROP TABLE `StoryAttribute`;
DROP TABLE `CharacterAttribute`;

CREATE TABLE Genre
(
	ID int(11) AUTO_INCREMENT NOT NULL,
	Type varchar(255) NOT NULL,
	PRIMARY KEY (ID),
	UNIQUE (Type)
);

CREATE TABLE Story
(
	ID int(11) AUTO_INCREMENT NOT NULL,
	Title varchar(255) NOT NULL,
	GenreID int(11) NOT NULL,
	PRIMARY KEY (ID),
	FOREIGN KEY (genreID) REFERENCES Genre(ID) ON UPDATE CASCADE,
	UNIQUE (Title)
);

CREATE TABLE OriginStory
(
	ID int(11) AUTO_INCREMENT NOT NULL,
	Type varchar(255) NOT NULL,
	PRIMARY KEY(ID)
);

CREATE TABLE CharacterArchetype
(
	ID int(11) AUTO_INCREMENT NOT NULL,
	Type varchar(255) NOT NULL,
```

```
	PRIMARY KEY(ID),
	UNIQUE (Type)
);

CREATE TABLE CharacterData
(
	ID int(11) AUTO_INCREMENT NOT NULL,
	FirstName varchar(255) NOT NULL,
	LastName varchar(255),
	StoryID int(11) NOT NULL,
	OriginStoryID int(11) NOT NULL,
	PRIMARY KEY(ID),
	FOREIGN KEY (StoryID) REFERENCES Story(ID) ON UPDATE CASCADE,
	FOREIGN KEY (OriginStoryID) REFERENCES OriginStory(ID) ON UPDATE CASCADE
);

CREATE TABLE CharacterIs
(
	CharacterID int(11) NOT NULL,
	ArchetypeID int(11) NOT NULL,
	PRIMARY KEY(CharacterID,ArchetypeID),
	FOREIGN KEY(CharacterID) REFERENCES CharacterData(ID) ON UPDATE
CASCADE ON DELETE CASCADE,
	FOREIGN KEY(ArchetypeID) REFERENCES CharacterArchetype(ID) ON UPDATE
CASCADE
);

-- The following queries are used to populate tables for the purpose of drop-down menus on
-- project site, since there were some menus that needed to know the column name.

CREATE TABLE StoryAttribute
(
	ID int(11) AUTO_INCREMENT NOT NULL,
	StoryAttribute varchar(255) NOT NULL,
	PRIMARY KEY(ID),
	UNIQUE(StoryAttribute)
);

CREATE TABLE CharacterAttribute
(
	ID int(11) AUTO_INCREMENT NOT NULL,
	CharacterAttribute varchar(255) NOT NULL,
	PRIMARY KEY(ID),
```

```
        UNIQUE(CharacterAttribute)
);

INSERT INTO StoryAttribute(StoryAttribute) VALUES('Story');
INSERT INTO StoryAttribute(StoryAttribute) VALUES('Genre');
INSERT INTO StoryAttribute(StoryAttribute) VALUES('Origin Story');

INSERT INTO CharacterAttribute(CharacterAttribute) VALUES('Number of Characters');
INSERT INTO CharacterAttribute(CharacterAttribute) VALUES('Character Archetype');
INSERT INTO CharacterAttribute(CharacterAttribute) VALUES('Origin Story');
```

General Use Queries

I want to see all of the queries that will be used to select, update, add or delete data. Because many of these will be based on user input, use square brackets to act as place holders for variables that will be user provided. For example, if I were going to query based on employee salaries, I might have a query like this:
SELECT salary FROM employee WHERE salary > [salaryInput];
Another example:
INSERT INTO employee(name, age) VALUES ([user],[name]);

```
-- From project.php, displays entire list of characters
SELECT CharacterData.FirstName, CharacterData.LastName, OriginStory.Type,
CharacterArchetype.Type, Story.Title FROM CharacterData
INNER JOIN OriginStory ON OriginStory.ID = CharacterData.OriginStoryID
INNER JOIN Story ON Story.ID = CharacterData.StoryID
INNER JOIN CharacterIs ON CharacterIs.CharacterID=CharacterData.ID
INNER JOIN CharacterArchetype ON CharacterArchetype.ID=CharacterIs.ArchetypeID"


-- From addArchetypeExisting, used to get CharacterData.ID for below INSERT
SELECT CharacterData.ID FROM CharacterData WHERE
CharacterData.FirstName=[InputFirstName] AND CharacterData.LastName=[InputLastName]
AND CharacterData.StoryID=[InputStoryID]

-- From addArchetypeExisting, CharacterID provided by above SELECT, Archetype provided by
-- user input
INSERT INTO CharacterIs (CharacterID,ArchetypeID) VALUES
([InputCharacterID],[InputArchetypeID])

-- From createNewArchetype
```

```
INSERT INTO CharacterArchetype (Type) VALUES ([InputNewArchetypeType])


-- From createNewCharacter, after creation newly generated ID will be used to insert value into
-- CharacterIs table
INSERT INTO CharacterData (FirstName,LastName,StoryID,OriginStoryID) VALUES
([inputFirstName],[inputLastName],[inputStoryID],[inputOrginStoryID])

-- From createNewCharacter, uses ID created from above INSERT to assign archetype to
proper -- character
INSERT INTO CharacterIs (CharacterID,ArchetypeID) VALUES
([inputCharacterIDFromAbove],[inputCharacterArchetypeID])

-- From createNewGenre
INSERT INTO Genre (Type) VALUES ([inputNewGenreType])

-- From createNewOriginStory
INSERT INTO OriginStory (Type) VALUES ([inputOriginStoryType])

-- From createNewStory
INSERT INTO Story (Title,GenreID) VALUES ([inputNewStoryTitle],[]inputNewStoryGenreID)

-- From deleteCharacter
DELETE FROM CharacterData WHERE CharacterData.FirstName=[inputCharacterFirstName]
AND CharacterData.LastName=[inputCharacterLastName] AND
CharacterData.StoryID=[inputStoryID]

-- From listByArchetype
SELECT CharacterData.FirstName, CharacterData.LastName, CharacterArchetype.Type,
OriginStory.Type, Story.Title
                FROM CharacterData
                INNER JOIN OriginStory ON OriginStory.ID = CharacterData.OriginStoryID
                INNER JOIN Story ON Story.ID = CharacterData.StoryID
                INNER JOIN CharacterIs ON CharacterIs.CharacterID=CharacterData.ID
                INNER JOIN CharacterArchetype ON
CharacterIs.ArchetypeID=CharacterArchetype.ID
                WHERE CharacterArchetype.ID=[inputArchetypeID]

-- From listByGenre
SELECT Genre.Type, Story.Title
                FROM Genre
                INNER JOIN Story ON Story.GenreID = Genre.ID
                WHERE Genre.ID=[inputGenreID]
```

```
-- From listByOriginStory
SELECT CharacterData.FirstName, CharacterData.LastName, CharacterArchetype.Type,
OriginStory.Type, Story.Title
		FROM CharacterData
		INNER JOIN OriginStory ON OriginStory.ID = CharacterData.OriginStoryID
		INNER JOIN Story ON Story.ID = CharacterData.StoryID
		INNER JOIN CharacterIs ON CharacterIs.CharacterID=CharacterData.ID
		INNER JOIN CharacterArchetype ON
CharacterIs.ArchetypeID=CharacterArchetype.ID
		WHERE OriginStory.ID=[inputOriginStoryID]

-- From listByStory
SELECT CharacterData.FirstName, CharacterData.LastName, CharacterArchetype.Type,
OriginStory.Type, Story.Title
		FROM CharacterData
		INNER JOIN OriginStory ON OriginStory.ID = CharacterData.OriginStoryID
		INNER JOIN Story ON Story.ID = CharacterData.StoryID
		INNER JOIN CharacterIs ON CharacterIs.CharacterID=CharacterData.ID
		INNER JOIN CharacterArchetype ON
CharacterIs.ArchetypeID=CharacterArchetype.ID
		WHERE Story.ID=[inputStoryID]

-- From searchByName, used if user doesn't input a LastName to search with
SELECT CharacterData.FirstName, CharacterData.LastName, CharacterArchetype.Type,
OriginStory.Type, Story.Title
		FROM CharacterData
		INNER JOIN OriginStory ON OriginStory.ID = CharacterData.OriginStoryID
		INNER JOIN Story ON Story.ID = CharacterData.StoryID
		INNER JOIN CharacterIs ON CharacterIs.CharacterID=CharacterData.ID
		INNER JOIN CharacterArchetype ON
CharacterIs.ArchetypeID=CharacterArchetype.ID
		WHERE CharacterData.FirstName = [inputFirstName]

-- From searchByName, used if user inputs FirstName and LastName
SELECT CharacterData.FirstName, CharacterData.LastName, CharacterArchetype.Type,
OriginStory.Type, Story.Title
		FROM CharacterData
		INNER JOIN OriginStory ON OriginStory.ID = CharacterData.OriginStoryID
		INNER JOIN Story ON Story.ID = CharacterData.StoryID
		INNER JOIN CharacterIs ON CharacterIs.CharacterID=CharacterData.ID
		INNER JOIN CharacterArchetype ON
CharacterIs.ArchetypeID=CharacterArchetype.ID
```

```
                WHERE CharacterData.FirstName = [inputFirstName] AND
CharacterData.LastName = [inputLastName]
-- From sumAttribute, used if storyAttribute is Genre and characterAttribute is
CharacterArchetype
"SELECT Genre.Type, ArchetypeCounts.Type,ArchetypeCounts.Counts FROM Genre
                        INNER JOIN Story ON Story.GenreID=Genre.ID
                        INNER JOIN (SELECT CharacterArchetype.Type,
COUNT(CharacterData.ID)AS `Counts`, CharacterData.StoryID FROM CharacterArchetype
                        INNER JOIN CharacterIs ON
CharacterIs.ArchetypeID=CharacterArchetype.ID
                        INNER JOIN CharacterData ON
CharacterData.ID=CharacterIs.CharacterID
                        GROUP BY CharacterArchetype.Type) AS ArchetypeCounts ON
ArchetypeCounts.StoryID=Story.ID
                        ORDER BY Genre.Type Asc;

-- From sumAttribute, used if storyAttribute is Genre and characterAttribute is Number of
-- Characters
SELECT Genre.Type,COUNT(CharacterData.ID) FROM CharacterData
                        INNER JOIN Story ON CharacterData.StoryID=Story.ID
                        INNER JOIN Genre ON Genre.ID=Story.GenreID
                        GROUP BY Genre.Type;

-- From sumAttribute, used if storyAttribute is Genre and characterAttribute is Origin Story
SELECT Genre.Type, OriginStoryCounts.Type,OriginStoryCounts.Counts FROM Genre
                        INNER JOIN Story ON Story.GenreID=Genre.ID
                        INNER JOIN (SELECT OriginStory.Type, COUNT(CharacterData.ID)AS
`Counts`, CharacterData.StoryID FROM OriginStory
                        INNER JOIN CharacterData ON
CharacterData.OriginStoryID=OriginStory.ID
                        GROUP BY OriginStory.Type) AS OriginStoryCounts ON
OriginStoryCounts.StoryID=Story.ID
                        ORDER BY Genre.Type Asc;

-- From sumAttribute, used if storyAttribute is Origin Story and characterAtribute is
-- CharacterArchetype
SELECT OriginStory.Type, ArchetypeCounts.Type,ArchetypeCounts.Counts FROM OriginStory
                        INNER JOIN CharacterData ON
CharacterData.OriginStoryID=OriginStory.ID
                        INNER JOIN (SELECT CharacterArchetype.Type,
COUNT(CharacterData.ID)AS `Counts`, CharacterData.ID FROM CharacterArchetype
                        INNER JOIN CharacterIs ON
CharacterIs.ArchetypeID=CharacterArchetype.ID
```

```
                        INNER JOIN CharacterData ON
CharacterData.ID=CharacterIs.CharacterID
                        GROUP BY CharacterArchetype.Type) AS ArchetypeCounts ON
ArchetypeCounts.ID=CharacterData.ID
                        ORDER BY OriginStory.Type Asc;

--From sumAttribute, used if storyAttribute is Origin Sotry and characterAttribute is Number of
-- Characters
SELECT OriginStory.Type,COUNT(CharacterData.ID) FROM CharacterData
                        INNER JOIN OriginStory ON CharacterData.OriginStoryID=OriginStory.ID
                        GROUP BY OriginStory.Type;

--From sumAttribute, used if storyAttribute is Story and character Attribute is Character
Archeytpe
SELECT Story.Title, ArchetypeCounts.Type,ArchetypeCounts.Counts FROM Story
                        INNER JOIN CharacterData ON CharacterData.StoryID=Story.ID
                        INNER JOIN (SELECT CharacterArchetype.Type,
COUNT(CharacterData.ID)AS `Counts`, CharacterData.ID FROM CharacterArchetype
                        INNER JOIN CharacterIs ON
CharacterIs.ArchetypeID=CharacterArchetype.ID
                        INNER JOIN CharacterData ON
CharacterData.ID=CharacterIs.CharacterID
                        GROUP BY CharacterArchetype.Type) AS ArchetypeCounts ON
ArchetypeCounts.ID=CharacterData.ID
                        ORDER BY Story.Title Asc;

--From sumAttribute, used if storyAttribute is Story and characterAttribute is Number of
-- Characters
SELECT Story.Title,COUNT(CharacterData.ID) FROM CharacterData
                        INNER JOIN Story ON CharacterData.StoryID=Story.ID
                        GROUP BY Story.Title;

--From sumAttribute, used if storyAttribute is Story and characterAttribute is Origin Story
SELECT Story.Title, OriginStoryCounts.Type,OriginStoryCounts.Counts FROM Story
                        INNER JOIN (SELECT OriginStory.Type, COUNT(CharacterData.ID)AS
`Counts`, CharacterData.StoryID FROM OriginStory
                        INNER JOIN CharacterData ON
CharacterData.OriginStoryID=OriginStory.ID
                        GROUP BY OriginStory.Type) AS OriginStoryCounts ON
OriginStoryCounts.StoryID=Story.ID
                        ORDER BY Story.Title Asc;
```