Nicolas Sim
CS 340

Final Project - Simple Maplestory database

Outline:

Maplestory is an online game that features characters that can travel across a virtual world while killing monsters and gaining levels. With a huge amount of detail put into this game which includes, but not limited to, joining a guild, defeating dungeons, quest-lines, and much more. In maplestory, one must create a character and choose a unique name. Once you begin the game, you are automatically placed as a “beginner” job at level 1. When time passes and you kill monsters you have the ability to choose a job at level 10. You may choose 1 out of 4 jobs, or choose not to advance from beginner at all. There are up to 5 rank ups throughout the maximum level cap - 250. Each job title has a hometown, and you have the ability to teleport to that hometown every hour. This game has become more complex over the years, including up to 32 different classes from its original 4.

Database Outline:

This database highlights the most important relationships in order to create a basic character in maplestory.

Character -

Character ID	name	rank	level	hometown	job

The Character table’s primary key is it’s ID. The character class has a name, a rank, and a level. It also has a hometown foreign key as well as a job foreign key.

Relationships -
Characters has a job: The character must have a job. All Characters participate in this relationship showing total participation
Character has a hometown: The character must have a hometown. All Characters participate in this relationship showing total participation
Character kills Boss(es): Each Character kills 0 or more Bosses. And a Boss is killed by 0 or more Characters. Total participation is not apparent in this relationship because a character does not have to kill a boss, and a boss does not have to be killed by a character.

Hometown -

Hometown ID	name	leader	description

The Hometown’s primary key is it’s ID. The Hometown class has a name, a leader and a description of the hometown. Has no foreign keys.

Relationships -
Hometown has characters from it: The hometown might not have any characters from it, but all characters must have a hometown. All Characters participate in this relationship, but not all hometowns participate in it. This is a one to many relationship as well. One hometown - many characters.
Hometown has bosses from it: The hometown might not have any monsters from it, but all monster have a hometown. All Bosses participate in this relationship, but not all hometowns participate in it. This is a one to many relationship as well. One hometown - many bosses.

Job -

Job ID	name	weapon

The Job's primary key is it's ID. The job class has a name and a weapon associated with that job.
Relationships -
Job has Characters in it: Each Character can only have 1 job at a time, but each Job can have many Characters in it. It is required for a Character to have a job, but it is not required for a job to have characters. Character has total participation in this relationship. This is a one to many relationship.

Boss -

Boss ID	name	level	town	hp

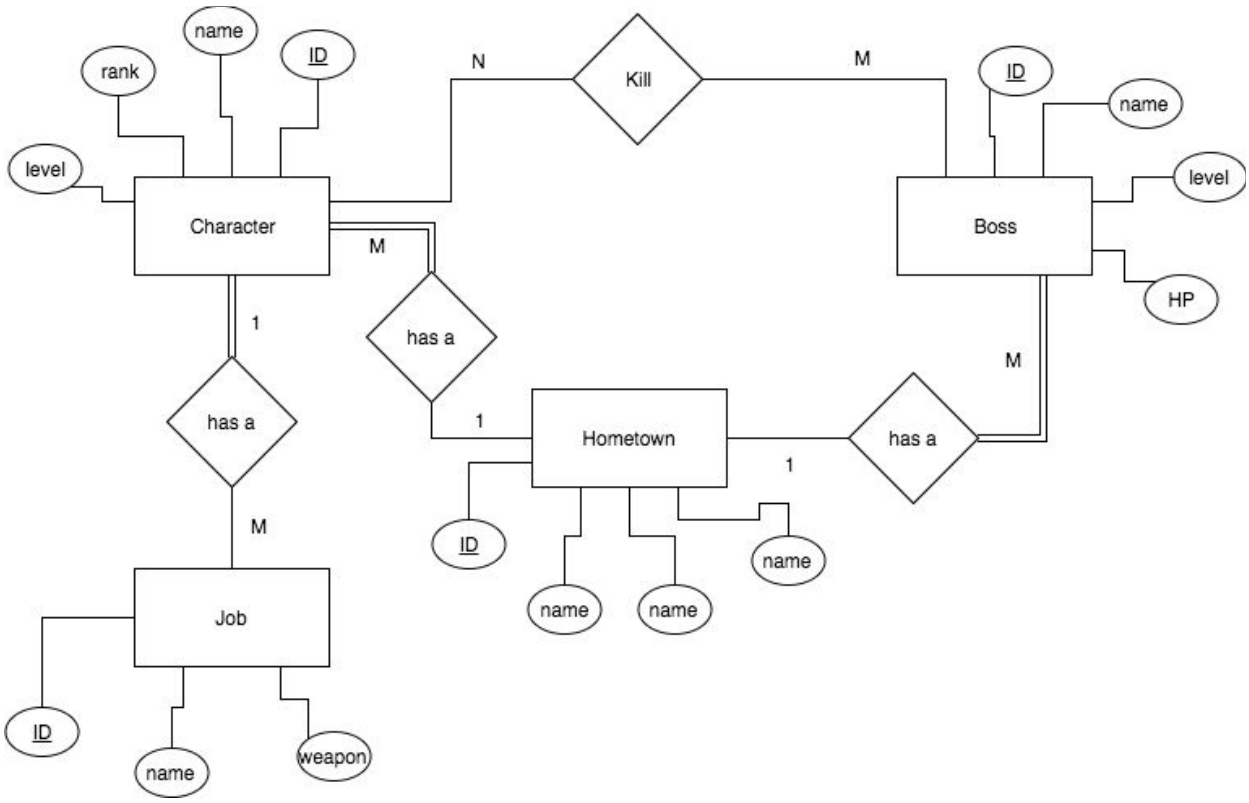
The Boss's primary key is it's ID. The Boss has a name, a level, a town where it's from and its health points (hp). The town is a foreign key linking to a town in the hometown table.
Relationships -
Bosses are killed by many characters: Many Characters are killed by bosses, and many characters kill bosses. This is a many to many relationship. Not all Bosses participate, and not all Characters have to participate. There is no total participation in this relationship.
Bosses have a hometown: All monsters have a hometown. Not all hometowns have a monster. All bosses participate in this relationship - total participation. One to many relationship.

Kill -

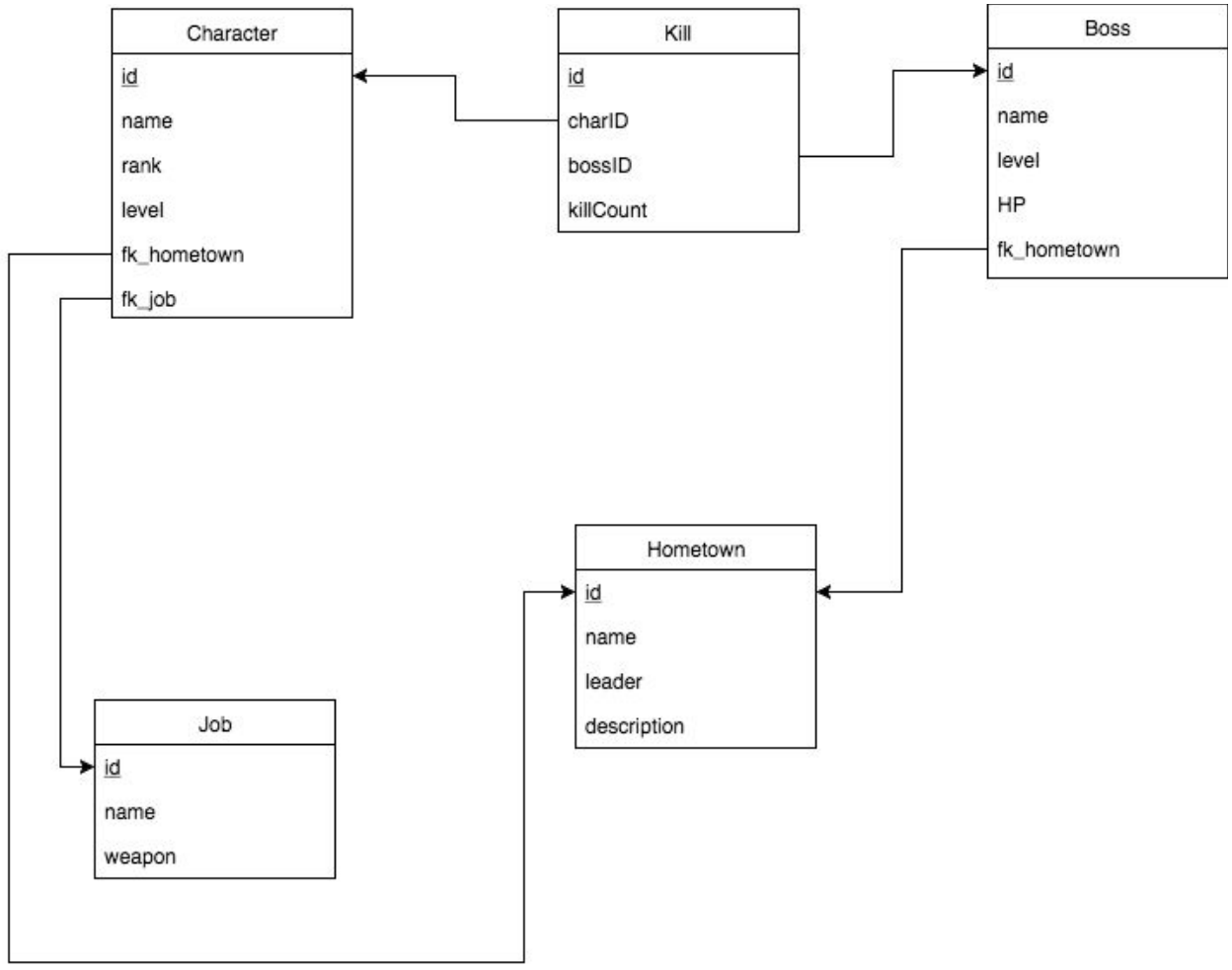
Kill ID	Character ID	Boss ID	killCount

The kill table is a junction table. Kill ID is it's primary key. Character ID and Boss ID are it's foreign keys. It keeps track of each Character and each boss as well as the killCount between the two foreign keys. Has a many to many relationship between Character and Boss because many characters can kill many bosses, and many bosses can be killed by many characters. But not all Characters must kill a boss, nor does a Boss need to be killed by a character.

ER Diagram:


name
rank
ID
level
Character
N
Kill
M
ID
name
level
Boss
HP
M
1
has a
has a
1
Hometown
has a
M
1
M
ID
name
name
name
Job
ID
name
weapon

Schema:


Character
id
name
rank
level
fk_hometown
fk_job
Kill
id
charID
bossID
killCount
Boss
id
name
level
HP
fk_hometown
Hometown
id
name
leader
description
Job
id
name
weapon

Data Definition Queries:

```
CREATE TABLE `character` (
        id INT AUTO_INCREMENT NOT NULL,
        name VARCHAR(255) NOT NULL,
        rank INT,
        level INT,
        fk_hometown INT NOT NULL,
        fk_job INT NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (fk_hometown) REFERENCES hometown(id),
        FOREIGN KEY (fk_job) REFERENCES job(id)
) ENGINE=InnoDB

CREATE TABLE `hometown` (
        id INT AUTO_INCREMENT NOT NULL,
        name VARCHAR(255) NOT NULL,
        leader VARCHAR(255),
        description VARCHAR(255),
        PRIMARY KEY (id)
) ENGINE=InnoDB

CREATE TABLE `boss` (
        id INT AUTO_INCREMENT NOT NULL,
        name VARCHAR(255) NOT NULL,
        hp INT,
        level INT,
        fk_hometown INT,
        PRIMARY KEY (id),
        FOREIGN KEY (fk_hometown) REFERENCES hometown(id)
) ENGINE=InnoDB

CREATE TABLE `job` (
        id INT AUTO_INCREMENT NOT NULL,
        name VARCHAR(255) NOT NULL,
        weapon VARCHAR(255),
        PRIMARY KEY (id)
        ) ENGINE=InnoDB
```

```
CREATE TABLE `kill` (
        id INT AUTO_INCREMENT NOT NULL,
        killCount INT,
        fk_charID INT NOT NULL,
        fk_bossID INT NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (fk_charID) REFERENCES character(id),
        FOREIGN KEY (fk_bossID) REFERENCES boss(id)
) ENGINE=InnoDB
```

Data Manipulation Queries:

```
SELECT c.name, j.weapon
FROM `character`
INNER JOIN `job` j ON c.fk_job = j.id
----//This is for use in finding what type of weapon your character can use

SELECT c.name, h.name
FROM `character` c
INNER JOIN `hometown` h ON c.fk_hometown = h.id
----//This is for use in finding what hometown your character can teleport to

SELECT c.name, j.name
FROM `character` c
INNER JOIN `job` j ON c.fk_job = j.id
----//This is for use in finding what job your character is

SELECT b.name, h.name
FROM `boss` b
INNER JOIN `hometown` h ON b.fk_hometown = h.id
----//This is for use in querying what city each monster resides at.

SELECT c.name, b.name, k.killCount
FROM `kill` k
INNER JOIN `character` c ON c.id = k.fk_charID
INNER JOIN `boss` b ON b.id = k.fk_bossID
WHERE killCount > 0
----//This is for use in querying what bosses your character has defeated, and if so, how many time.
```