Yong Lee

Final Project.

- Outline

My final project is about a MOBA game called SMITE which is like League of Legend except it's a third-person view game. An each player takes on a character in the game and fights the other team until one of the teams emerges victorious. Every character always derives from a mythology and has a distinct physical appearance, and they all belong to a specific role(class) though some can play another role depending on players' choices in game.

This database for someone wishes to look up characters depending on his preference whether that preference is a role, mythology, physical appearance, character attribute or rivalry, or for someone who wants to keep track of most current data as the game changes and updates with each patch.

- Database Outline

This database has 5 tables.

1) Character table - id, name, health, mana, mythology id (fk), physical appearance id (fk), enemyid (fk from its own table) === (some attributes are voided from this database)

2) Mythology table - id, name, origin

3) Physical Appearance Table - name, popularity (ranked from 1st to so on, data from a poll taken awhile back from a fan site, there can be a tie)

4) Role Table - id, name, range type

5) Character-Role Relationship table - character id (fk), role id (fk)

There are 4 relationships.

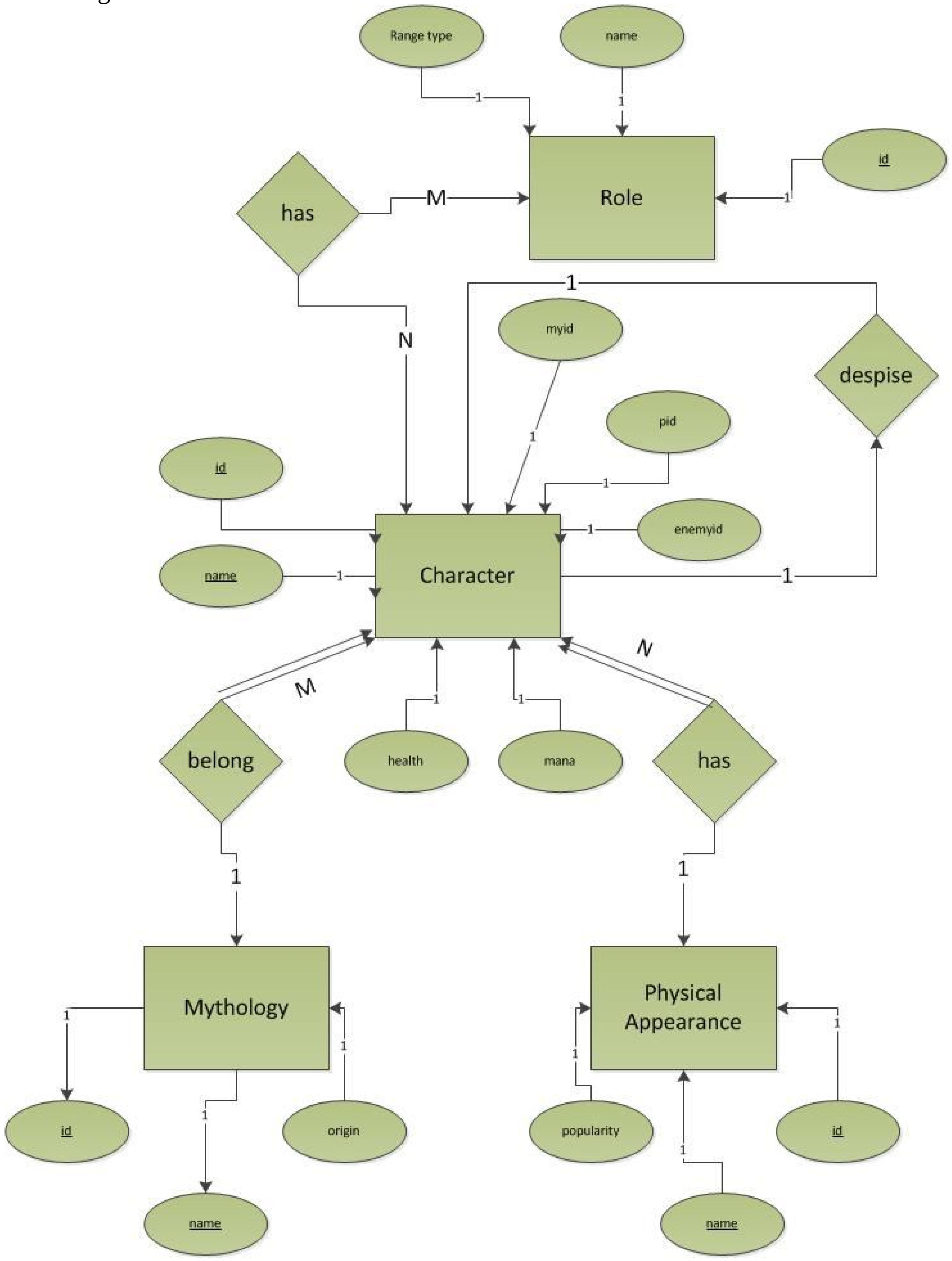
1) A character must derive from a mythology. - A character <u>must</u> only belong to one mythology. This is a one to many relationship.

2) A character must have physical appearance. - A character <u>must</u> have only one dominant physical appearance; therefore this is a one to many relationship.

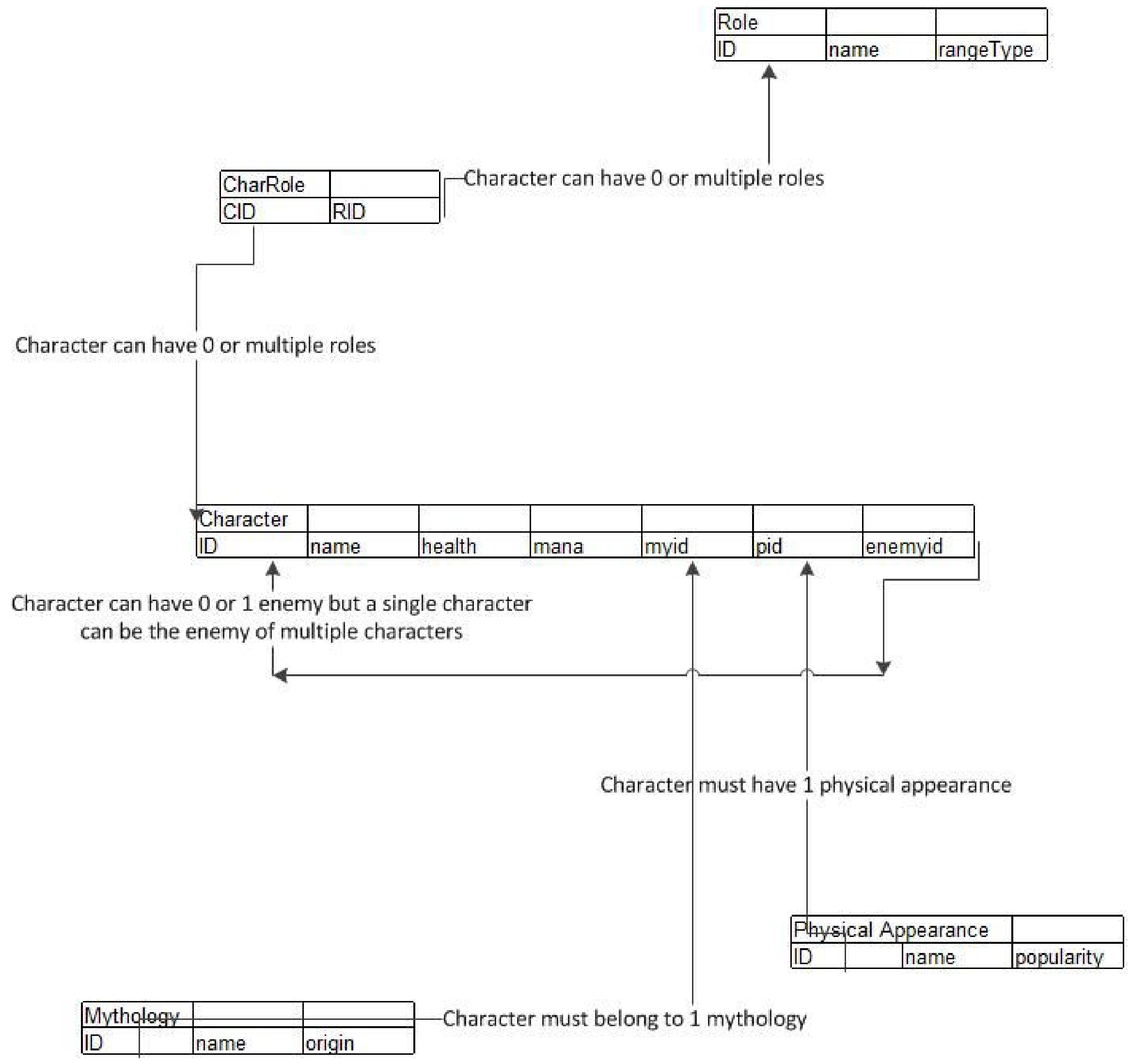
3) A character can belong to a role. - Some characters can fulfill few different roles while others can‘t, so this is a many to many relationship. A character can be without roles.

4) A character can have a sworn enemy. - A character may have one enemy that he despises or none. Being the enemy does not have go both ways and multiple characters can despise a single character.

- ER Diagram


Range type
name
1
1
Role
id
1
has
M
N
1
despise
myid
1
pid
1
id
enemyid
1
Character
name
1
1
M
N
belong
health
1
mana
1
has
1
1
Mythology
Physical Appearance
id
1
1
origin
name
1
popularity
1
id
1
name
1

- Database Schema


Role
ID
name
rangeType
CharRole
CID
RID
Character can have 0 or multiple roles
Character can have 0 or multiple roles
Character
ID
name
health
mana
myid
pid
enemyid
Character can have 0 or 1 enemy but a single character can be the enemy of multiple characters
Character must have 1 physical appearance
Physical Appearance
ID
name
popularity
Mythology
ID
name
origin
Character must belong to 1 mythology

- Table Creation Queries

```
-- Characters --
CREATE TABLE 275Character (
  id INT AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  health INT NOT NULL,
  mana INT NOT NULL,
  myid INT NOT NULL,
  pid INT NOT NULL,
  enemyid INT,
  PRIMARY KEY  (id),
  UNIQUE KEY (name),
  FOREIGN KEY (myid) REFERENCES 275Mythology (id) ON DELETE CASCADE
ON UPDATE CASCADE,
  FOREIGN KEY (pid) REFERENCES 275PhysicalApp (id) ON DELETE CASCADE
ON UPDATE CASCADE,
  FOREIGN KEY (enemyid) REFERENCES 275Character (id) ON DELETE CASCADE
ON UPDATE CASCADE
)ENGINE=InnoDB;

-- Mythology --
CREATE TABLE 275Mythology (
  id INT AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  origin VARCHAR(255) NOT NULL,
  PRIMARY KEY  (id),
  UNIQUE KEY(name)
)ENGINE=InnoDB;

-- Physical Appearance --
CREATE TABLE 275PhysicalApp (
  id INT AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  popularity VARCHAR(255) NOT NULL,
  PRIMARY KEY  (id),
  UNIQUE KEY(name)
)ENGINE=InnoDB;

-- Roles --
CREATE TABLE 275Role (
  id INT AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  rangeType VARCHAR(255) NOT NULL,
  PRIMARY KEY  (id),
  UNIQUE KEY(name)
)ENGINE=InnoDB;

-- Characters-Roles, Only Many to Many relationship table --
CREATE TABLE 275CharRole (
  cid INT NOT NULL,
  rid INT NOT NULL,
  PRIMARY KEY  (cid, rid),
  FOREIGN KEY (cid) REFERENCES 275Character (id) ON DELETE CASCADE ON
UPDATE CASCADE,
  FOREIGN KEY (rid) REFERENCES 275Role (id) ON DELETE CASCADE ON
UPDATE CASCADE
)ENGINE=InnoDB;
```

- General Use Queries

Showing current table

+Character - SELECT id, name, health, mana, myid, pid, enemyid FROM 275Character ORDER BY id

+Mythology - SELECT id, name, origin FROM 275Mythology ORDER BY id

+Physical Appearance - SELECT id, name, popularity FROM 275PhysicalApp ORDER BY id

+Role - SELECT id, name, rangeType FROM 275Role ORDER BY id

+Character-Role - SELECT cid, rid FROM 275CharRole ORDER BY cid

** Thought about using select command for foreign key, but that seems redundant since I already used it to get the ID in the first place.

Add

+Character - INSERT INTO 275Character(name, health, mana, myid, pid, enemyid) VALUES ([nameInput], [healthInput], [manaInput], [mythologyidInput], [physicalappearanceInput], [enemyidInput])
--If There is a character-role relationship
SELECT id from 275Character WHERE name=[nameInput]
INSERT INTO 275CharRole(cid, rid) VALUES ([characteridInput], [roleidInput])

+Mythology - INSERT INTO 275Mythology(name, origin) VALUES ([nameInput], [originInput])

+Physical Appearance - INSERT INTO 275PhysicalApp (name, popularity) VALUES ([nameInput], [popularityInput])

+Role - INSERT INTO 275Role (name, rangeType) VALUES ([nameInput], [rangeTypeInput)

Update

+Character - UPDATE 275Character SET health = [healthInput], mana= [manaInput] WHERE id = [idInput]

+Mythology - UPDATE 275Mythology SET origin = [originInput] WHERE id = [idInput]

+Physical Appearance - UPDATE 275PhysicalApp SET popularity = [popularityInput] WHERE id = [idInput]

+Role - UPDATE 275Role SET rangeType = [rangeInput] WHERE id = [idInput]

+Enemy - UPDATE 275Character SET enemyid = [enemyidInput] WHERE id = [idInput]

Relationship
+Relationship (character and mythology) - UPDATE 275Character SET myid = [mythidInput] WHERE id = [idInput]

+Relationship (character and physical appearance) - UPDATE 275Character SET pid = [physidInput] WHERE id = [idInput]

+Relationship (character and Role) (add only) - SELECT cid FROM 275CharRole WHERE cid=[characteridInput] AND rid=[roleidINput] (to check if this relationship already exist]

INSERT INTO 275CharRole(rid, cid) VALUES ([roleidINput], [characteridInput])

+Relationship (character and Role) (add only) - SELECT cid FROM 275CharRole WHERE cid=[cidInput] AND rid=[ridInput] (to check if this relationship already exist]

DELETE FROM 275CharRole WHERE cid=[characteridInput] AND rid=[roleidINput] (to check if this relationship already exist]

Delete

+Character - DELETE FROM 275Character WHERE id = [idInput]

+Mythology - DELETE FROM 275Mythology WHERE id = [idInput]

+Physical Appearance - DELETE FROM 275PhysicalApp WHERE id = [idInput]

+Role - DELETE FROM 275Role WHERE id = [idInput]

Select

Select 1 - SELECT r.name, c.name, health, mana, myid, pid, enemyid FROM 275Character c INNER JOIN 275CharRole cr ON c.id = cr.cid INNER JOIN 275Role r ON cr.rid = r.id ORDER BY r.name

Select 2 - SELECT c.name, health, mana, myid, pid, enemyid, p.name, r.name FROM 275Character c INNER JOIN 275CharRole cr ON c.id = cr.cid INNER JOIN 275Role r ON cr.rid = r.id INNER JOIN 275PhysicalApp p ON p.id = c.pid WHERE p.id =[PhysicalAppidInput] AND r.id =[rolidInput] ORDER BY c.name

Select 3 - SELECT c.name, health, mana, myid, pid, enemyid, m.name, r.name FROM 275Character c INNER JOIN 275CharRole cr ON c.id = cr.cid INNER JOIN 275Role r ON cr.rid = r.id INNER JOIN 275Mythology m ON m.id = c.myid WHERE m.id =[mythologyidInput] AND NOT r.id=[roleidINput] ORDER BY c.name

Select 4 - SELECT count(c.name) FROM 275Character c INNER JOIN 275PhysicalApp p ON c.pid = p.id WHERE c.health > [healthInput] AND c.mana < [manaInput] AND p.id = [phyidInput]

Select 5 - SELECT sum(c.health), sum(c.mana) FROM 275Character c INNER JOIN 275PhysicalApp p ON c.pid = p.id INNER JOIN 275Mythology m ON m.id = c.myid WHERE m.id = [MythologyidInput] AND NOT p.id = [PhysicalAppidInput]